10/723,603

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年１１月２５日

出願番号
Application Number: 特願２００３－３９３４１６

[ST. 10/C]: [ＪＰ２００３－３９３４１６]

出願人
Applicant(s): 株式会社リコー

２００３年１２月１８日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００３－３１０５０７８

【書類名】　特許願
【整理番号】　0308738
【提出日】　平成15年11月25日
【あて先】　特許庁長官　今井　康夫　殿
【国際特許分類】　G03G 21/00
　　G06F 17/60
【発明者】
　【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
　【氏名】　小林　綾子
【特許出願人】
　【識別番号】　000006747
　【氏名又は名称】　株式会社リコー
【代理人】
　【識別番号】　100070150
　【弁理士】
　【氏名又は名称】　伊東　忠彦
【先の出願に基づく優先権主張】
　【出願番号】　特願2002-342826
　【出願日】　平成14年11月26日
【手数料の表示】
　【予納台帳番号】　002989
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　特許請求の範囲　1
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　9911477

【書類名】特許請求の範囲
【請求項１】
画像形成処理で使用されるハードウェア資源と、画像形成に係る処理を行うプログラムとを有する画像形成装置であって、
前記ハードウェア資源に関するチェックを行うチェック処理と前記プログラムとの関連を設定する設定手段と、
前記チェック処理を行い、そのチェック処理の結果に応じて前記チェック処理に関連する前記プログラムを起動する起動手段と
を有することを特徴とする画像形成装置。
【請求項２】
前記設定手段は、前記チェック処理と前記プログラムとを１対１に関連付けて設定していることを特徴とする請求項１記載の画像形成装置。
【請求項３】
前記設定手段は、前記チェック処理と前記プログラムとを１対ｎに関連付けて設定していることを特徴とする請求項１記載の画像形成装置。
【請求項４】
前記設定手段は、前記チェック処理と前記プログラムとをｎ対１に関連付けて設定していることを特徴とする請求項１記載の画像形成装置。
【請求項５】
前記チェック処理の結果を格納しておく格納手段を更に有し、
前記起動手段は、前記ハードウェア資源に関するチェックを行う前に前記格納手段に格納されているチェックの結果を確認し、これから行うチェックの結果が格納済みであれば前記チェックの結果を利用することを特徴とする請求項１記載の画像形成装置。
【請求項６】
前記起動手段は、前記設定手段に設定されている内容に応じて前記チェック処理を行うことを特徴とする請求項２乃至５何れか一記載の画像形成装置。
【請求項７】
前記起動手段は、前記プログラムを起動した後に、前記チェック処理を終了させることを特徴とする請求項６記載の画像形成装置。
【請求項８】
前記起動手段は、チェック処理を行って前記ハードウェア資源の有無をチェックし、前記ハードウェア資源が有ると判定したときに正常値をチェックの結果として取得し、前記ハードウェア資源が無いと判定したときに異常値をチェックの結果として取得することを特徴とする請求項２乃至７何れか一項記載の画像形成装置。
【請求項９】
前記起動手段は、前記チェック処理として前記ハードウェア資源に対応したデバイスドライバをオープンする為の処理を行い、オープンが成功したとき又は既にオープンされていたときに前記ハードウェア資源が有ると判定し、それ以外のときに前記ハードウェア資源がないと判定することを特徴とする請求項８記載の画像形成装置。
【請求項１０】
前記起動手段は、プリンタ，コピー，ファックス又はスキャナの一部又は全部として動作するハードウェア資源の有無のチェックの結果として正常値を取得したときに、前記設定手段に設定されている内容に応じてプリンタ，コピー，ファックス又はスキャナの処理を行う前記プログラムを起動することを特徴とする請求項８又は９記載の画像形成装置。
【請求項１１】
前記起動手段は、チェック処理を行って前記ハードウェア資源の有無をチェックし、前記ハードウェア資源が無いと判定したときに正常値をチェックの結果として取得し、前記ハードウェア資源が有ると判定したときに異常値をチェックの結果として取得することを特徴とする請求項２乃至７何れか一項記載の画像形成装置。
【請求項１２】

前記起動手段は、ハードディスク装置の有無のチェックの結果として正常値を取得したときに、前記設定手段に設定されている内容に応じてＲＡＭディスクをマウントすることを特徴とする請求項１１記載の画像処理装置。

【請求項１３】

前記起動手段は、前記チェック処理として前記ハードウェア資源の所定の性能をチェックし、前記ハードウェア資源が所定の性能を満たしていると判定したときに正常値をチェックの結果として取得し、前記ハードウェア資源が所定の性能を満たしていないと判定したときに異常値をチェックの結果として取得することを特徴とする請求項２乃至７何れか一項記載の画像形成装置。

【請求項１４】

前記起動手段は、中央演算処理装置（ＣＰＵ）の性能のチェックの結果として正常値を取得したときに、前記設定手段に前記チェック処理と関連付けて設定されている前記プログラムを起動し、前記チェックの結果として異常値を取得したときに、前記設定手段に前記チェック処理と関連付けて設定されている前記プログラムを起動しないことを特徴とする請求項１３記載の画像形成装置。

【請求項１５】

前記起動手段は、メモリの性能のチェックの結果として正常値を取得したときに、前記設定手段に前記チェック処理と関連付けて設定されている前記プログラムを起動し、前記チェックの結果として異常値を取得したときに、前記設定手段に前記チェック処理と関連付けて設定されている前記プログラムを起動しないことを特徴とする請求項１３記載の画像形成装置。

【請求項１６】

前記設定手段は、前記チェック処理と前記プログラムの存在するディレクトリ又は上位のディレクトリとを関連付けて設定しており、

前記起動手段は、前記チェックの結果が正常値であるときに前記チェック処理に関連付けて設定されている前記ディレクトリをマウントし、前記チェックの結果が異常値であるときに前記チェック処理に関連付けて設定されている前記ディレクトリをマウントしないことを特徴とする請求項１４又は１５記載の画像形成装置。

【請求項１７】

前記起動手段は、チェック処理を行って前記ハードウェア資源に格納されている所定の識別子をチェックし、前記所定の識別子が所定の条件を満たしていると判定したときに正常値をチェックの結果として取得し、前記所定の識別子が所定の条件を満たしていないと判定したときに異常値をチェックの結果として取得することを特徴とする請求項２乃至７何れか一項記載の画像形成装置。

【請求項１８】

前記起動手段は、チェック処理を行ってＳＤカードに格納されている識別子をチェックし、その識別子が前記ＳＤカードが挿入されているスロットの識別子と一致していると判定したときに正常値をチェックの結果として取得し、前記識別子が前記ＳＤカードが挿入されているスロットの識別子と一致していないと判定したときに異常値をチェックの結果として取得することを特徴とする請求項１７記載の画像形成装置。

【請求項１９】

前記起動手段は、前記チェックの結果として正常値を取得したときに、前記設定手段に前記チェック処理と関連付けて設定されている前記プログラムを実行し、前記チェックの結果として異常値を取得したときに、前記設定手段に前記チェック処理と関連付けて設定されている前記プログラムを実行しないことを特徴とする請求項１７又は１８記載の画像形成装置。

【請求項２０】

前記格納手段は、前記起動手段が直接アクセス可能なメモリ上の領域に作成されることを特徴とする請求項５記載の画像形成装置。

【請求項２１】

前記起動手段は、電源投入後に起動されたオペレーティングシステムにより起動されることを特徴とする請求項１記載の画像形成装置。

【請求項２２】

前記プログラムは、画像形成処理を行うアプリケーションのプログラムと、画像形成処理で利用されるハードウェア資源の管理を行うコントロールサービスのプログラムと、オペレーティングシステムのプログラムとを有することを特徴とする請求項１記載の画像形成装置。

【請求項２３】

画像形成処理で使用されるハードウェア資源と、画像形成に係る処理を行うプログラムとを有する画像形成装置のプログラム起動方法であって、

起動手段が、前記プログラムと前記ハードウェア資源に関するチェックを行うチェック処理との関連を設定した設定手段を解析する解析段階と、

前記起動手段が、前記解析の結果に応じて前記チェック処理を実行するチェック処理実行段階と、

前記起動手段が、前記チェック処理の結果に応じて前記チェック処理に関連する前記プログラムを起動するプログラム起動段階と

を有することを特徴とするプログラム起動方法。

【請求項２４】

画像形成処理で使用されるハードウェア資源と、画像形成に係る処理を行うプログラムとを有するコンピュータを、

前記ハードウェア資源に関するチェックを行うチェック処理と前記プログラムとの関連を設定する設定手段と、

前記チェック処理を行い、そのチェック処理の結果に応じて前記チェック処理に関連する前記プログラムを起動する起動手段と

して機能させるためのプログラム起動プログラム。

【書類名】明細書
【発明の名称】画像形成装置，プログラム起動方法及びプログラム起動プログラム
【技術分野】
【０００１】
本発明は、画像形成装置，プログラム起動方法及びプログラム起動プログラムに係り、特に所定の設定ファイルに応じてプログラムを起動するプログラム起動方法，そのプログラム起動方法を利用する画像形成装置及びプログラム起動プログラムに関する。
【背景技術】
【０００２】
近年、プリンタ，コピー，ファクシミリおよびスキャナなどの各装置の機能を１つの筐体内に収納した画像形成装置（以下、融合機という）が知られるようになった。この融合機は、１つの筐体内に表示部，印刷部および撮像部などを設けると共に、プリンタ，コピー，ファクシミリおよびスキャナにそれぞれ対応する４種類のソフトウェアを設け、そのソフトウェアを切り替えることより、プリンタ，コピー，ファクシミリおよびスキャナとして動作させるものである。例えば特許文献１には、融合機の一例が記載されている。
【０００３】
このような融合機は、電源投入後に、ＢＩＯＳ（Basic Input/Output System）およびブートローダ（Boot Loader）が起動する。ブートローダは、カーネル（Kernel）およびルートファイルシステムをＲＡＭ（Random Access Memory）上に展開してカーネルを起動する。そして、カーネルはルートファイルシステムをマウントする。ここでマウントとは、ファイルシステムや周辺機器などをアクセス可能な状態に起動することをいう。
【０００４】
カーネルの起動後、アプリケーション（以下、アプリという）や各種サービスを起動するアプリ／サービス層起動プログラムが起動される。アプリ／サービス層起動プログラムは融合機で最初に起動されるプロセスであり、所定の設定ファイルに従ってファイルシステムをマウントし、融合機の動作に必要なサービス層およびアプリ層のプロセスを所定の設定ファイルに従って起動している。従来の融合機では、起動されたサービス層およびアプリ層のプロセスが、各プロセスの処理の中で表示部，印刷部および撮像部などのハードウェア資源のチェックを行っていた。例えば特許文献２には、予め定義された動作定義情報に従いアプリを起動するプログラムの一例が記載されている。
【特許文献１】特開２００２－８４３８３号公報
【特許文献２】特開２０００－２０２０３号公報
【発明の開示】
【発明が解決しようとする課題】
【０００５】
しかしながら、従来の融合機ではハードウェア資源を複数のプロセスで共通に利用しており、各プロセスの処理の中でハードウェア資源のチェックを行うと各プロセスに重複した部分が多くなるという問題があった。また、従来の融合機では各プロセスの処理の中でハードウェア資源のチェックを行うため、各プロセスを起動しなければハードウェア資源の有無，性能などのチェックを行うことができなかった。
【０００６】
したがって、従来の融合機ではハードウェア資源が存在しない，ハードウェア資源の性能が低い等の理由で使用しないプロセスであっても、ハードウェア資源のチェックを行うために無駄に起動しなければならないという問題があった。なお、特許文献２はハードウェア資源のチェックを行うものでなく、上記の問題を解決することができない。
【０００７】
本発明は、上記の点に鑑みなされたもので、各プログラムの重複部分を削減することができ、ハードウェア資源に関連するプログラムを効率良く起動することが可能な画像形成装置，プログラム起動方法及びプログラム起動プログラムを提供することを目的とする。
【課題を解決するための手段】

【０００８】
そこで、上記課題を解決するため、本発明は、画像形成処理で使用されるハードウェア資源と、画像形成に係る処理を行うプログラムとを有する画像形成装置であって、前記ハードウェア資源に関するチェックを行うチェック処理と前記プログラムとの関連を設定する設定手段と、前記チェック処理を行い、そのチェック処理の結果に応じて前記チェック処理に関連する前記プログラムを起動する起動手段とを有することを特徴とする。
【０００９】
また、本発明は、画像形成処理で使用されるハードウェア資源と、画像形成に係る処理を行うプログラムとを有する画像形成装置のプログラム起動方法であって、起動手段が、前記プログラムと前記ハードウェア資源に関するチェックを行うチェック処理との関連を設定した設定手段を解析する解析段階と、前記起動手段が、前記解析の結果に応じて前記チェック処理を実行するチェック処理実行段階と、前記起動手段が、前記チェック処理の結果に応じて前記チェック処理に関連する前記プログラムを起動するプログラム起動段階とを有することを特徴とする。
【００１０】
また、本発明は、画像形成処理で使用されるハードウェア資源と、画像形成に係る処理を行うプログラムとを有するコンピュータを、前記ハードウェア資源に関するチェックを行うチェック処理と前記プログラムとの関連を設定する設定手段と、前記チェック処理を行い、そのチェック処理の結果に応じて前記チェック処理に関連する前記プログラムを起動する起動手段として機能させるためのプログラム起動プログラムであることを特徴とする。
【００１１】
本発明によれば、プログラムを起動する起動手段がハードウェア資源に関するチェックを行うようにしたため、各プログラムの処理の中でハードウェア資源に関するチェックを行う必要が無くなり、各プログラムの重複部分を削減できる。
【００１２】
また、本発明によれば、ハードウェア資源に関するチェックの結果に応じてプログラムを起動するため、使用しないプログラムを起動する必要がなくなり、プログラムを効率良く起動できる。
【発明の効果】
【００１３】
上述の如く、本発明によれば、各プログラムの重複部分を削減することができ、ハードウェア資源に関連するプログラムを効率良く起動することが可能な画像形成装置，プログラム起動方法及びプログラム起動プログラムを提供できる。
【発明を実施するための最良の形態】
【００１４】
次に、本発明を実施するための最良の形態を、以下の実施例に基づき図面を参照しつつ説明していく。
【００１５】
図１は、本発明による融合機のソフトウェア構成について説明するための一実施例の構成図である。融合機１は、ソフトウェア群２と，融合機起動部３と，ハードウェア資源４とを含むように構成される。
【００１６】
ハードウェア資源４は、プロッタ１１と，スキャナ１２と，ファクシミリなどのその他のハードウェアリソース１３とを含む。ソフトウェア群２は、ＵＮＩＸ（登録商標）などのオペレーティングシステム（以下、ＯＳという）上に起動されているアプリケーション層５とプラットフォーム６とを含む。
【００１７】
アプリケーション層５は、プリンタ，コピー，ファックスおよびスキャナなどの画像形成にかかるユーザサービスにそれぞれ固有の処理を行うプログラムを含む。図１のアプリ

ケーション層５は、プリンタアプリ２１と，コピーアプリ２２と，ファックスアプリ２３と，スキャナアプリ２４と，ネットファイルアプリ２５とを含む。なお、ネットファイルアプリ２５はネットワークファイル用アプリケーションであり、融合機１にネットワークを介して接続されるネットワーク機器とのデータ通信を管理するものである。

【００１８】

プラットフォーム６は、アプリケーション層５からの処理要求を解釈してハードウェア資源４の獲得要求を発生するコントロールサービス層９と、１つ以上のハードウェア資源４の管理を行ってコントロールサービス層９からの獲得要求を調停するシステムリソースマネージャ（以下、ＳＲＭという）３９と、ＳＲＭ３９からの獲得要求に応じてハードウェア資源４の管理を行うハンドラ層１０とを含む。

【００１９】

コントロールサービス層９は、ＮＣＳ３１，ＤＣＳ３２，ＯＣＳ３３，ＦＣＳ３４，ＥＣＳ３５，ＭＣＳ３６，ＵＣＳ３７，ＳＣＳ３８など、一つ以上のサービスモジュールを含む。なお、プラットフォーム６は予め定義されている関数により、アプリケーション層５からの処理要求を受信するＡＰＩ５３を有するように構成されている。ＯＳは、アプリケーション層５およびプラットフォーム６の各ソフトウェアをプロセスとして並列実行する。

【００２０】

ＮＣＳ（ネットワークコントロールサービス）３１のプロセスは、ネットワーク側から各プロトコルによって受信したデータを各アプリケーションに振り分ける際の仲介、又は各アプリケーションからのデータをネットワーク側に送信する際の仲介を行う。例えばＮＣＳ３１は、融合機にネットワークを介して接続されるネットワーク機器とのデータ通信を制御する。

【００２１】

ＤＣＳ（デリバリーコントロールサービス）３２のプロセスは、融合機に蓄積されている文書データの配送などの制御を行う。ＯＣＳ（操作パネルコントロールサービス）３３のプロセスは、後述する操作パネルの制御を行う。

【００２２】

ＦＣＳ（ファックスコントロールサービス）３４のプロセスは、アプリケーション層５からＰＳＴＮまたはＩＳＤＮ網を利用したファックスの送受信，バックアップ用のメモリで管理されている各種ファックスデータの登録又は引用，ファックスの読み取り，ファックスの受信印刷などを行うためのＡＰＩを提供する。

【００２３】

ＥＣＳ（エンジンコントロールサービス）３５のプロセスは、プロッタ１１，スキャナ１２，ハードウェアリソース１３などのエンジン部の制御を行う。ＭＣＳ（メモリコントロールサービス）３６のプロセスは、メモリの取得及び解放，ＨＤＤの利用，画像データの圧縮および伸張などの制御を行う。ＵＣＳ（ユーザ情報コントロールサービス）３７のプロセスは、ユーザ情報の管理を行うものである。

【００２４】

ＳＣＳ（システムコントロールサービス）３８のプロセスは、操作部の制御，システム画面の表示，ＬＥＤの表示，ハードウェア資源の管理，アプリケーションの管理，割り込みアプリケーションの制御などの処理を行う。

【００２５】

ＳＲＭ３９のプロセスは、ＳＣＳ３８と共にシステムの制御およびハードウェア資源４の管理を行うものである。例えばＳＲＭ３９のプロセスは、プロッタ１１やスキャナ１２などのハードウェア資源４を利用する上位層からの獲得要求に従って調停を行い、ハードウェア資源４の実行を制御する。

【００２６】

具体的に、ＳＲＭ３９のプロセスは獲得要求されたハードウェア資源４が利用可能であるか（他の獲得要求により利用されていないか）を判定し、利用可能であれば獲得要求さ

れたハードウェア資源４が利用可能である旨を上位層に通知する。ＳＲＭ３９のプロセスは、上位層からの獲得要求に対してハードウェア資源４を利用するためのスケジューリングを行い、要求内容（プリンタエンジンによる紙搬送と作像動作，メモリの確保，ファイル生成など）を直接実施している。

【００２７】

また、ハンドラ層１０は後述するＦＣＵ（ファックスコントロールユニット）の管理を行うＦＣＵＨ（ファックスコントロールユニットハンドラ）４０と、プロセスに対するメモリの割り振り及びプロセスに割り振ったメモリの管理を行うＩＭＨ（イメージメモリハンドラ）４１とを含む。ＳＲＭ３９及びＦＣＵＨ４０は、予め定義されている関数によりハードウェア資源４に対する処理要求を送信するエンジンＩ／Ｆ５４を利用して、ハードウェア資源４に対する処理要求を行う。

【００２８】

図１の構成により、融合機１は各アプリケーションで共通的に必要な処理をプラットフォーム６で一元的に処理することができる。次に、融合機１のハードウェア構成について説明する。

【００２９】

図２は、本発明による融合機のハードウェア構成について説明するための一実施例の構成図である。図２の融合機１は、コントローラ６０，操作パネル８０，ＦＣＵ８１，エンジン部８２を有する。

【００３０】

コントローラ６０は、ＣＰＵ６１，システムメモリ６２，ＮＢ６３，ＳＢ６４，ＡＳＩＣ６６，ローカルメモリ６７，ＨＤＤ６８、ＮＩＣ６９，ＳＤカード用スロット７０，ＵＳＢ　Ｉ／Ｆ７１，ＩＥＥＥ１３９４　Ｉ／Ｆ７２，セントロニクス　Ｉ／Ｆ７３を有する。

【００３１】

操作パネル８０は、コントローラ６０のＡＳＩＣ６６に接続されている。また、ＦＣＵ８１およびエンジン部８２はコントローラ６０のＡＳＩＣ６６にＰＣＩバス８３を介して接続されている。

【００３２】

コントローラ６０は、ＡＳＩＣ６６にローカルメモリ６７，ＨＤＤ６８などが接続されると共に、ＣＰＵ６１とＡＳＩＣ６６とがＣＰＵチップセットのＮＢ６３を介して接続されている。なお、ＡＳＩＣ６６とＮＢ６３とはＡＧＰ（Accelerated Graphics Port）６５を介して接続されている。

【００３３】

ＣＰＵ６１は、融合機１の全体制御を行うものである。図１の融合機１では、ＣＰＵ６１がコントロールサービス層９を形成する１つ以上のサービスモジュールと、ＳＲＭ３９と、ハンドラ層１０を形成するＦＣＵＨ４０，ＩＭＨ４１とをＯＳ上に起動させた後、アプリケーション層５を形成するプリンタアプリ２１，コピーアプリ２２，ファックスアプリ２３，スキャナアプリ２４，ネットファイルアプリ２５を起動して実行させる。

【００３４】

ＮＢ（ノースブリッジ）６３は、ＣＰＵ６１，システムメモリ６２，ＳＢ６４，ＡＳＩＣ６６，ＮＩＣ６９，ＳＤカード用スロット７０，ＵＳＢ　Ｉ／Ｆ７１，ＩＥＥＥ１３９４　Ｉ／Ｆ７２及びセントロニクス　Ｉ／Ｆ７３を接続するためのブリッジである。ＮＢ６３は、ＳＢ６４，ＮＩＣ６９，ＳＤカード用スロット７０，ＵＳＢ　Ｉ／Ｆ７１，ＩＥＥＥ１３９４　Ｉ／Ｆ７２及びセントロニクス　Ｉ／Ｆ７３とＰＣＩバス７４を介して接続されている。なお、ＳＢ（サウスブリッジ）６４は、ＰＣＩバス７４とＲＯＭや周辺デバイス等とを接続するためのブリッジである。

【００３５】

システムメモリ６２は、描画用メモリ等として用いるメモリである。ローカルメモリ６７は、コピー用画像バッファ，符号バッファ等として用いるメモリである。ＡＳＩＣ６６

は、画像処理用のハードウェア要素を有する画像処理用途向けのＩＣである。また、ＨＤＤ６８は画像データの蓄積，文書データの蓄積，プログラムの蓄積，フォントデータの蓄積，フォームの蓄積などを行うストレージ（補助記憶装置）の一例である。

【００３６】

ＮＩＣ（ネットワークインターフェースカード）６９は、融合機１をインターネットやＬＡＮ等のネットワークに接続するインターフェース機器である。ＳＤカード用スロット７０はＳＤカードを挿抜可能なものであり、ＳＤカードの挿入または抜き出しに応じた割り込みをデバイスドライバに対して行う。

【００３７】

ＵＳＢ　Ｉ／Ｆ７０，ＩＥＥＥ１３９４　Ｉ／Ｆ７１およびセントロニクス　Ｉ／Ｆ７２は、夫々の規格に準じたインターフェースである。操作パネル８０は、操作者からの入力操作を受け付けると共に、操作者に向けた表示を行う操作部である。なお、ＦＣＵ８１はバックアップ用のメモリを有している。ＦＣＵ８１が有するメモリは、例えば融合機１の電源がＯＦＦのときに受信したファクシミリデータを一時的に格納するために利用される。

【００３８】

図３は、融合機起動部の一例の構成図である。図１の融合機起動部３は、融合機１の電源投入時に最初に実行され、アプリケーション層５やプラットフォーム６を起動するものである。融合機起動部３は、ＲＯＭモニタ５１と，プログラム起動部５２とを含む。融合機起動部３の処理について図４のフローチャートを参照しつつ説明する。

【００３９】

図４は、融合機起動部の処理の一例のフローチャートである。ステップＳ１０１では、融合機１の電源ＯＮにより、ＢＩＯＳおよびブートローダとしてのＲＯＭモニタ５１が実行される。ＲＯＭモニタ５１は、ハードウェアの初期化，コントローラ６０の診断，ソフトウェアの初期化などを行う。ステップＳ１０１に続いてステップＳ１０２に進み、ＲＯＭモニタ５１は、ＯＳおよびルートファイルシステムをシステムメモリ６２上に展開してＯＳを起動する。そして、ＯＳはルートファイルシステムをマウントする。

【００４０】

ステップＳ１０２に続いてステップＳ１０３に進み、ＯＳは起動時に接続されたデバイスのデバイス情報（例えば、ＣＰＵ６１のクロック周波数，システムメモリ６２およびローカルメモリ５７のメモリサイズ，コントローラ６０のボードタイプなど）を取得する。

【００４１】

ステップＳ１０３に続いてステップＳ１０４に進み、ＯＳはアプリ／サービス起動プログラムとしてのプログラム起動部５２を起動する。プログラム起動部５２はシステムメモリ６２およびローカルメモリ６７上にメモリ領域を確保する。プログラム起動部５２は、融合機１で最初に起動されるプロセスである。ステップＳ１０４に続いてステップＳ１０５に進み、プログラム起動部５２は設定ファイルに従ってファイルシステムをマウントする。

【００４２】

また、プログラム起動部５２は設定ファイルに従って、ハードウェア資源に関するチェックを行うチェック処理を実行する。プログラム起動部５２は、チェック処理が正常に終了したか否かで、設定ファイルに記述されたアプリケーション層５およびプラットフォーム６のプログラム（以下、本体プログラムという）を起動するか否かを判定する。

【００４３】

本体プログラムを起動すると判定した場合、プログラム起動部５２は、その本体プログラムを設定ファイルに従ってＲＯＭなどから読み出し、読み出した本体プログラムをシステムメモリ６２，ローカルメモリ６７上に確保したメモリ領域に展開して、アプリケーション層５およびプラットフォーム６のプロセスを起動する。以下、プログラム起動部５２が行うステップＳ１０５の処理について、詳細に説明していく。

【実施例１】

【００４４】
図５は、プログラム起動部の処理の一例のフローチャートである。まず、ステップＳ１１０では、プログラム起動部５２が設定ファイルを解析する。ステップＳ１１０に続いてステップＳ１１１に進み、プログラム起動部５２は設定ファイルに従ってファイルシステムをマウントする。
【００４５】
ステップＳ１１１に続いてステップＳ１１２に進み、プログラム起動部５２は設定ファイルに記述されたｅｘｅｃコマンドを１つ読み出し、そのｅｘｅｃコマンドに「－ｈ」オプションがあるか否かを判定する。例えば図６のような設定ファイルの場合、プログラム起動部５２は１行目のｅｘｅｃコマンドに「－ｈ」オプションがあると判定する。ｅｘｅｃコマンドに「－ｈ」オプションがあると判定すると（Ｓ１１２においてＹＥＳ）、プログラム起動部５２はステップＳ１１３に進み、ｅｘｅｃコマンドで指定されたチェック処理を実行する。例えば図６のような設定ファイルの場合、プログラム起動部５２は１行目のｅｘｅｃコマンドで指定されたチェック処理「ｆｃｕｃｈｅｃｋ」を実行する。ステップＳ１１３のチェック処理では、ハードウェア資源に関するチェック（例えばハードウェア資源の有無，性能などのチェック）を行い、チェックの結果を取得する。
【００４６】
ステップＳ１１３に続いてステップＳ１１４に進み、プログラム起動部５２は実行したチェック処理の結果に基づいて、チェック処理が正常終了したか異常終了したかを判定する。チェック処理が正常終了したと判定すると（Ｓ１１４においてＹＥＳ）、プログラム起動部１４はステップＳ１１５に進む。
【００４７】
ステップＳ１１５では、プログラム起動部５２が、ｅｘｅｃコマンドで指定された本体プログラムを起動する。例えば図６のような設定ファイルの場合、プログラム起動部５２は１行目のｅｘｅｃコマンドで指定された本体プログラム「／ｆａｘ／ｂｉｎ／ｆａｘ」を起動する。この後、プログラム起動部１４はチェック処理を終了する。
【００４８】
ステップＳ１１５に続いてステップＳ１１６に進み、プログラム起動部５２は起動すべき本体プログラム，言い換えれば読み込んでいないｅｘｅｃコマンドが設定ファイルにまだ存在するか否かを判定する。まだ読み込んでいないｅｘｅｃコマンドが設定ファイルに存在すると判定すると（Ｓ１１６においてＹＥＳ）、プログラム起動部５２はステップＳ１１２に戻り、まだ読み込んでいないｅｘｅｃコマンドを設定ファイルから読み込んでステップＳ１１２以降の処理を行う。
【００４９】
一方、まだ読み込んでいないｅｘｅｃコマンドが設定ファイルに存在しないと判定すると（Ｓ１１６においてＮＯ）、プログラム起動部５２は処理を終了する。なお、ステップＳ１１２において、ｅｘｅｃコマンドに「－ｈ」オプションがないと判定すると（Ｓ１１２においてＮＯ）、プログラム起動部５２はステップＳ１１５に進み、ｅｘｅｃコマンドで指定された本体プログラムを起動する。即ち、ｅｘｅｃコマンドに「－ｈ」オプションがなければ、プログラム起動部５２はチェック処理を実行することなくｅｘｅｃコマンドで指定された本体プログラムを起動する。
【００５０】
また、ステップＳ１１４において、チェック処理が正常終了していないと判定すると（Ｓ１１４においてＮＯ）、プログラム起動部５２はステップＳ１１６に進む。即ち、チェック処理が異常終了した場合、プログラム起動部５２はｅｘｅｃコマンドで指定された本体プログラムを起動しない。
【００５１】
図５のフローチャートの処理により、プログラム起動部５２はチェック処理が正常終了したときにｅｘｅｃコマンドで指定された本体プログラムを起動する一方、チェック処理が異常終了したときにｅｘｅｃコマンドで指定された本体プログラムを起動しないことに

より融合機１の動作を抑制できる。

【００５２】

なお、図５のフローチャートでは、コマンドの一例としてｅｘｅｃコマンドを例に説明したが、ｍｏｕｎｔコマンドであってもよい。この場合、プログラム起動部５２はチェック処理が正常終了したときにｍｏｕｎｔコマンドで指定されたマウントを行う一方、チェック処理が異常終了したときにｍｏｕｎｔコマンドで指定されたマウントを行わないことにより融合機１のマウント動作を抑制できる。

【００５３】

次に、図６の設定ファイルを用いて複数のチェック処理について説明していく。図６の設定ファイルでは、１行目のｅｘｅｃコマンドに「－ｈ」オプションがあるので、チェック処理「ｆｃｕｃｈｅｃｋ」が実行される。例えばプログラム起動部５２は、図７のようなチェック処理「ｆｃｕｃｈｅｃｋ」を行う。

【００５４】

図７は、チェック処理「ｆｃｕｃｈｅｃｋ」の一例のフローチャートである。ステップＳ１２０では、プログラム起動部５２がＦＣＵ８１のデバイスドライバをオープンする為の処理を行う。ステップＳ１２０に続いてステップＳ１２１に進み、プログラム起動部５２はステップＳ１２０でのデバイスドライバのオープンが成功したか否かを判定する。

【００５５】

デバイスドライバのオープンが成功したと判定すると（Ｓ１２１においてＹＥＳ）、プログラム起動部５２はステップＳ１２３に進み、ＦＣＵ８１が融合機１に接続されているとみなして、正常終了を表すチェック結果を取得する。一方、デバイスドライバのオープンが失敗したと判定すると（Ｓ１２１においてＮＯ）、プログラム起動部５２はステップＳ１２２に進み、ＦＣＵ８１が既にビジー状態であるか否かを判定する。なお、ＦＣＵ８１がビジー状態であるか否かは、例えば「ｅｒｒｎｏ」に「ＥＢＵＳＹ」が入っているか否かで確認できる。

【００５６】

ＦＣＵ８１が既にビジー状態であると判定すると（Ｓ１２２においてＹＥＳ）、プログラム起動部５２はステップＳ１２４に進み、ＦＣＵ８１が融合機１に接続されているとみなして、正常終了を表すチェック結果を取得する。一方、ＦＣＵ８１が既にビジー状態であると判定しなかった場合（Ｓ１２２においてＮＯ）、プログラム起動部５２はステップＳ１２５に進み、ＦＣＵ８１が融合機１に接続されていないとみなして、異常終了を表すチェック結果を取得する。

【００５７】

図７のフローチャートの処理により、プログラム起動部５２はＦＣＵ８１が融合機１に接続されている場合に正常終了を表すチェック結果を取得し、ＦＣＵ８１が融合機１に接続されていない場合に異常終了を表すチェック結果を取得できる。なお、プログラム起動部５２は正常終了を表すチェック結果を取得したときにアプリ「ｆａｘ」を起動し、異常終了を表すチェック結果を取得したときにアプリ「ｆａｘ」を起動しない。

【００５８】

したがって、プログラム起動部５２は正常終了または異常終了を表すチェック結果を利用して、ＦＣＵ８１が融合機１に接続されている場合にアプリ「ｆａｘ」を起動し、接続されていない場合にアプリ「ｆａｘ」を起動しないようにすることができる。即ち、プログラム起動部５２はアプリ「ｆａｘ」の起動をＦＣＵ８１の接続状態によって抑制できる。なお、図７のフローチャートでは、デバイスの一例としてのＦＣＵ８１を用いて説明したが、如何なるデバイスであってもよい。つまり、図７のフローチャートによれば、オプションボード等のデバイスの接続状態によってアプリの起動を抑制することができる。

【００５９】

図６の設定ファイルでは、１行目のｅｘｅｃコマンドの処理が終了すると、２行目のｅｘｅｃコマンドの処理が行われる。図６の設定ファイルでは、２行目のｅｘｅｃコマンドに「－ｈ」オプションがあるので、チェック処理「ｃｐｕｃｈｅｃｋ１」が実行される。

例えばプログラム起動部５２は、図８のようなチェック処理「ｃｐｕｃｈｅｃｋ１」を行う。

【００６０】

図８は、チェック処理「ｃｐｕｃｈｅｃｋ１」の一例のフローチャートである。ステップＳ１３０では、プログラム起動部５２がシステムコール「ｇｅｔＩＮＦＯ（ＣＰＵ）」を呼び、デバイス情報に含まれているＣＰＵ６１のクロック周波数をＯＳから取得する。

【００６１】

ステップＳ１３０に続いてステップＳ１３１に進み、プログラム起動部５２はステップＳ１３０で取得したＣＰＵ６１のクロック周波数が５００ＭＨｚ以下であるか否かを判定する。ＣＰＵ６１のクロック周波数が５００ＭＨｚ以下であると判定すると（Ｓ１３１においてＹＥＳ）、プログラム起動部５２はステップＳ１３２に進み、正常終了を表すチェック結果を取得する。一方、ＣＰＵ６１のクロック周波数が５００ＭＨｚ以下でないと判定すると（Ｓ１３１においてＮＯ）、プログラム起動部５２はステップＳ１３３に進み、異常終了を表すチェック結果を取得する。

【００６２】

図８のフローチャートの処理により、プログラム起動部５２はＣＰＵ６１のクロック周波数が５００ＭＨｚ以下である場合に正常終了を表すチェック結果を取得し、ＣＰＵ６１のクロック周波数が５００ＭＨｚ以下でない場合に異常終了を表すチェック結果を取得できる。なお、プログラム起動部５２は正常終了を表すチェック結果を取得したときにアプリ「ｓｅｔｆｏｎｔ＿ｂｉｔｍａｐ」を起動し、異常終了を表すチェック結果を取得したときにアプリ「ｓｅｔｆｏｎｔ＿ｂｉｔｍａｐ」を起動しない。

【００６３】

したがって、プログラム起動部５２は正常終了または異常終了を表すチェック結果を利用して、ＣＰＵ６１のクロック周波数が５００ＭＨｚ以下である場合にプリンタが使用するデフォルトフォントをビットマップ形式に設定する。即ち、プログラム起動部５２はＣＰＵ６１のクロック周波数が５００ＭＨｚ以下であっても、一定時間内にプリントできるようなデフォルトフォントに変えることで、プリントの高速性を優先できる。

【００６４】

図６の設定ファイルでは、２行目のｅｘｅｃコマンドの処理が終了すると、３行目のｅｘｅｃコマンドの処理が行われる。図６の設定ファイルでは、３行目のｅｘｅｃコマンドに「－ｈ」オプションがあるので、チェック処理「ｃｐｕｃｈｅｃｋ２」が起動される。例えばプログラム起動部５２は、図９のようなチェック処理「ｃｐｕｃｈｅｃｋ２」を行う。

【００６５】

図９は、チェック処理「ｃｐｕｃｈｅｃｋ２」の一例のフローチャートである。ステップＳ１４０では、プログラム起動部５２がシステムコール「ｇｅｔＩＮＦＯ（ＣＰＵ）」を呼び、デバイス情報に含まれているＣＰＵ６１のクロック周波数をＯＳから取得する。

【００６６】

ステップＳ１４０に続いてステップＳ１４１に進み、プログラム起動部５２はステップＳ１４０で取得したＣＰＵ６１のクロック周波数が５０１ＭＨｚ以上であるか否かを判定する。ＣＰＵ６１のクロック周波数が５０１ＭＨｚ以上であると判定すると（Ｓ１４１においてＹＥＳ）、プログラム起動部５２はステップＳ１４２に進み、正常終了を表すチェック結果を取得する。一方、ＣＰＵ６１のクロック周波数が５０１ＭＨｚ以上でないと判定すると（Ｓ１４１においてＮＯ）、プログラム起動部５２はステップＳ１４３に進み、異常終了を表すチェック結果を取得する。

【００６７】

図９のフローチャートの処理により、プログラム起動部５２はＣＰＵ６１のクロック周波数が５０１ＭＨｚ以上である場合に正常終了を表すチェック結果を取得し、ＣＰＵ６１のクロック周波数が５０１ＭＨｚ以上でない場合に異常終了を表すチェック結果を取得できる。なお、プログラム起動部５２は、正常終了を表すチェック結果を取得したときにア

プリ「ｓｅｔｆｏｎｔ＿ｖｅｃｔｏｒ」を起動し、異常終了を表すチェック結果を取得したときにアプリ「ｓｅｔｆｏｎｔ＿ｖｅｃｔｏｒ」を起動しない。

【００６８】

したがって、プログラム起動部５２は正常終了または異常終了を表すチェック結果を利用して、ＣＰＵ６１のクロック周波数が５０１ＭＨｚ以上である場合にプリンタが使用するデフォルトフォントをベクトル画像に設定する。即ち、プログラム起動部５２はＣＰＵ６１のクロック周波数が５０１ＭＨｚ以上であるときに、精細なデフォルトフォントに変えることで、プリントの高画質化が可能である。

【００６９】

図８および図９のフローチャートの処理により、本発明の融合機１は一定の時間内にプリントできるように、ＣＰＵ６１の性能に応じて用いるデフォルトフォントを変えることで、プリントの高速性を優先することができる。

【００７０】

図６の設定ファイルでは、３行目のｅｘｅｃコマンドの処理が終了すると、４行目のｅｘｅｃコマンドの処理が行われる。図６の設定ファイルでは、４行目のｅｘｅｃコマンドに「－ｈ」オプションがあるので、チェック処理「ｍｅｍｃｈｅｃｋ１」が実行される。例えばプログラム起動部５２は、図１０のようなチェック処理「ｍｅｍｃｈｅｃｋ１」を行う。

【００７１】

図１０は、チェック処理「ｍｅｍｃｈｅｃｋ１」の一例のフローチャートである。ステップＳ１５０では、プログラム起動部５２がシステムコール「ｇｅｔＩＮＦＯ（ｍｅｍ）」を呼び、デバイス情報に含まれるシステムメモリ６２及びローカルメモリ６７のメモリサイズをＯＳから取得する。ステップＳ１５０に続いてステップＳ１５１に進み、プログラム起動部５２はステップＳ１５０で取得したメモリサイズが６４ＭＢ以上１２８ＭＢ以下であるか否かを判定する。

【００７２】

メモリサイズが６４ＭＢ以上１２８ＭＢ以下であると判定すると（Ｓ１５１においてＹＥＳ）、プログラム起動部５２はステップＳ１５２に進み、正常終了を表すチェック結果を取得する。一方、メモリサイズが６４ＭＢ以上１２８ＭＢ以下でないと判定すると（Ｓ１５１においてＮＯ）、プログラム起動部５２はステップＳ１５３に進み、異常終了を表すチェック結果を取得する。

【００７３】

図１０のフローチャートの処理により、プログラム起動部５２はメモリサイズが６４ＭＢ以上１２８ＭＢ以下である場合に正常終了を表すチェック結果を取得し、メモリサイズが６４ＭＢ以上１２８ＭＢ以下でない場合に異常終了を表すチェック結果を取得できる。なお、プログラム起動部５２は正常終了を表すチェック結果を取得したときにｈｔｔｐのデーモン（以下、ｈｔｔｐｄという）を５個起動し、異常終了を表すチェック結果を取得したときにｈｔｔｐｄを起動しない。

【００７４】

したがって、プログラム起動部５２は正常終了または異常終了を表すチェック結果を利用することで、システムメモリ６２及びローカルメモリ６７のメモリサイズが小さい場合であっても、起動するｈｔｔｐｄの数を少なくして、一つの要求に対する処理時間の増大を防ぐことができる。

【００７５】

図６の設定ファイルでは、４行目のｅｘｅｃコマンドの処理が終了すると、５行目のｅｘｅｃコマンドの処理が行われる。図６の設定ファイルでは、５行目のｅｘｅｃコマンドに「－ｈ」オプションがあるので、チェック処理「ｍｅｍｃｈｅｃｋ２」が実行される。例えばプログラム起動部５２は、図１１のようなチェック処理「ｍｅｍｃｈｅｃｋ２」を行う。

【００７６】

図１１は、チェック処理「ｍｅｍｃｈｅｃｋ２」の一例のフローチャートである。ステップＳ１６０では、プログラム起動部５２がシステムコール「ｇｅｔＩＮＦＯ（ｍｅｍ）」を呼び、デバイス情報に含まれるシステムメモリ６２及びローカルメモリ６７のメモリサイズをＯＳから取得する。ステップＳ１６０に続いてステップＳ１６１に進み、プログラム起動部５２はステップＳ１６０で取得したメモリサイズが１２８ＭＢ以上であるか否かを判定する。

【００７７】

メモリサイズが１２８ＭＢ以上であると判定すると（Ｓ１６１においてＹＥＳ）、プログラム起動部５２はステップＳ１６２に進み、正常終了を表すチェック結果を取得する。一方、メモリサイズが１２８ＭＢ以上でないと判定すると（Ｓ１６１においてＮＯ）、プログラム起動部５２はステップＳ１６３に進み、異常終了を表すチェック結果を取得する。

【００７８】

図１１のフローチャートの処理により、プログラム起動部５２はメモリサイズが１２８ＭＢ以上である場合に正常終了を表すチェック結果を取得し、メモリサイズが１２８ＭＢ以上でない場合に異常終了を表すチェック結果を取得できる。なお、プログラム起動部５２は正常終了を表すチェック結果を取得したときにｈｔｔｐｄを１０個起動し、異常終了を表すチェック結果を取得したときにｈｔｔｐｄを起動しない。

【００７９】

したがって、プログラム起動部５２は正常終了または異常終了を表すチェック結果を利用することで、システムメモリ６２及びローカルメモリ６７のメモリサイズが大きい場合に、起動するｈｔｔｐｄの数を多くして、クライアントからの要求にすぐ対応することができる。

【００８０】

図１０および図１１のフローチャートの処理により、本発明の融合機１はシステムメモリ６２及びローカルメモリ６７のメモリサイズに応じて、起動するｈｔｔｐｄの数を適切に変えることができる。

【００８１】

次に、図１２の設定ファイルを用いてプログラム起動部５２のチェック処理について説明する。図１２の設定ファイルでは、ｍｏｕｎｔコマンドに「－ｈ」オプションがあるので、チェック処理「ｈｄｄｎｏｎｅｘｉｓｔ」が実行される。例えばプログラム起動部５２は、図１３のようなチェック処理「ｈｄｄｎｏｎｅｘｉｓｔ」を行う。

【００８２】

図１３は、チェック処理「ｈｄｄｎｏｎｅｘｉｓｔ」の一例のフローチャートである。ステップＳ１７０では、プログラム起動部５２がシステムコール「ｇｅｔＩＮＦＯ（ｈｄｄ）」を呼び、デバイス情報に含まれているＨＤＤ接続有無情報をＯＳから取得する。

【００８３】

ステップＳ１７０に続いてステップＳ１７１に進み、プログラム起動部５２はステップＳ１７１で取得したＨＤＤ接続有無情報に応じてＨＤＤが融合機１に接続されているか否かを判定する。ＨＤＤが融合機１に接続されていないと判定すると（Ｓ１７１においてＮＯ）、プログラム起動部５２はステップＳ１７２に進み、正常終了を表すチェック結果を取得する。一方、ＨＤＤが融合機１に接続されていると判定すると（Ｓ１７１においてＹＥＳ）、プログラム起動部５２はステップＳ１７３に進み、異常終了を表すチェック結果を取得する。

【００８４】

図１３のフローチャートの処理により、プログラム起動部５２はＨＤＤが融合機１に接続されていない場合に正常終了を表すチェック結果を取得し、ＨＤＤが融合機１に接続されている場合に異常終了を表すチェック結果を取得できる。なお、プログラム起動部５２は、正常終了を表すチェック結果を取得したときにｒａｍｄｉｓｋをマウントする。具体的には、「／ｄｅｖ／ｍｄ０ｃ」がマウントポイント「／ｒａｍｄｉｓｋ」へマウントさ

れる。また、プログラム起動部５２は異常終了を表すチェック結果を取得したときにｒａｍｄｉｓｋをマウントしない。

【００８５】

したがって、プログラム起動部５２は正常終了または異常終了を表すチェック結果を利用することで、ＨＤＤが融合機１に接続されていない場合にｒａｍｄｉｓｋをマウントする。即ち、プログラム起動部５２はＨＤＤが融合機１に接続されていない場合であっても、ＰＤＬストレージ用のローカルストレージ用デバイスとしてｒａｍｄｉｓｋを用いることができる。なお、プログラム起動部５２はＨＤＤが融合機１に接続されている場合、ＰＤＬストレージ用のローカルストレージ用デバイスとしてＨＤＤを用いることができる。

【００８６】

次に、図１４の設定ファイルを用いてプログラム起動部５２のチェック処理について説明する。図１４の設定ファイルは、図１５のようにＳＤカード内に格納されている。図１５に表されたＳＤカードの場合、「ｘｘｘ．ｃｎｆ」が設定ファイル、「ｍｏｄｕｌｅ／ｘｘｘ．ｍｏｄ」がマウント及び実行対象のモジュールファイルを表している。

【００８７】

図１４の設定ファイルでは、ｍｏｕｎｔコマンドに「－ｈ」オプションがあるので、チェック処理「ｓｄｃｏｍｍａｎｄ」が実行される。例えばプログラム起動部５２は、図１６のようなチェック処理「ｓｄｃｏｍｍａｎｄ」を行う。

【００８８】

図１６は、チェック処理「ｓｄｃｏｍｍａｎｄ」の一例のフローチャートである。ステップＳ１８０では、プログラム起動部５２がＳＤカード内の図１４のような設定ファイルを解析する。ステップＳ１８０に続いてステップＳ１８１に進み、プログラム起動部５２はステップＳ１８０の解析結果から設定ファイルにＳＤコマンドが存在するか否かを判定する。ＳＤコマンドが存在すると判定すると（Ｓ１８１においてＹＥＳ）、プログラム起動部５２はステップＳ１８２に進む。一方、ＳＤコマンドが存在しないと判定すると（Ｓ１８１においてＮＯ）、プログラム起動部５２はステップＳ１８３に進む。

【００８９】

ステップＳ１８２では、プログラム起動部５２が、ＳＤコマンドで指定されているスロットとＳＤカードが挿入されているスロットとが一致しているか否かを判定する。例えば図１４のような設定ファイルの場合、ＳＤコマンドで指定されているスロットが「２」であるため、ＳＤカードが挿入されているスロットが「２」であるときに、プログラム起動部５２はＳＤコマンドで指定されているスロットとＳＤカードが挿入されているスロットとが一致していると判定する。

【００９０】

ＳＤコマンドで指定されているスロットとＳＤカードが挿入されているスロットとが一致していると判定すると（Ｓ１８２においてＹＥＳ）、プログラム起動部５２はステップＳ１８３に進み、正常終了を表すチェック結果を取得する。一方、ＳＤコマンドで指定されているスロットとＳＤカードが挿入されているスロットとが一致していないと判定すると（Ｓ１８２においてＮＯ）、プログラム起動部５２はステップＳ１８４に進み、異常終了を表すチェック結果を取得する。

【００９１】

図１６のフローチャートの処理により、プログラム起動部５２はＳＤコマンドで指定されているスロットとＳＤカードが挿入されているスロットとが一致している場合に正常終了を表すチェック結果を取得し、ＳＤコマンドで指定されているスロットとＳＤカードが挿入されているスロットとが一致していない場合に異常終了を表すチェック結果を取得できる。

【００９２】

なお、プログラム起動部５２は正常終了を表すチェック結果を取得したときにｇｚｉｐ圧縮されたＲＯＭＦＳ形式のモジュールファイル「ｘｘｘ．ｍｏｄ」をマウントポイント「／ｍｎｔ」へマウントし、そのモジュールファイルを実行する。また、プログラム起動

部５２は異常終了を表すチェック結果を取得したときにモジュールファイルをマウント及び実行しない。

【００９３】

したがって、プログラム起動部５２は正常終了または異常終了を表すチェック結果を利用することで、ＳＤコマンドで指定されているスロットと異なるスロットに挿入されているＳＤカード内のモジュールファイルのマウント及び実行を抑制することができる。

【００９４】

図６の設定ファイルの場合、本体プログラムと，チェック処理と，プログラム起動部５２と，ＯＳと，ハードウェア資源との関係は、例えば図１７のようになる。図１７は、本体プログラムと，チェック処理と，プログラム起動部５２と，ＯＳと，ハードウェアとの関係を表した一例の関係図である。

【００９５】

プログラム起動部５２はチェック処理を順番に実行し、チェック処理が正常終了したとき、そのチェック処理に対応する本体プログラムを起動する。図１７では、各チェック処理の上に位置している本体プログラムがチェック処理に対応する本体プログラムである。

【実施例２】

【００９６】

図１７は、プログラム起動部５２のチェック処理と本体プログラムとを１対１に対応させた例を表しているが、図１８のようにチェック処理と本体プログラムとを１対ｎに対応させるようにしてもよい。

【００９７】

図１８は、チェック処理と本体プログラムとが１対ｎに対応する例について説明するための図である。図１８の例では、プログラム起動部５２のチェック処理ａと本体プログラムａ，ｂとが対応している。プログラム起動部５２はチェック処理ａを実行し、チェック処理ａが正常終了したとき、そのチェック処理ａに対応する本体プログラムａ，ｂを起動する。図１８のようにチェック処理ａと本体プログラムａ，ｂとが対応している場合、設定ファイルは図１９のように構成される。図１９は、設定ファイルの他の一例の構成図である。なお、プログラム起動部５２の処理は図５と同様であるので説明を省略する。

【００９８】

また、同じチェック処理によるチェックの結果に基づき、ディレクトリ下の複数の本体プログラムを起動するような場合、チェックの結果に基づいてディレクトリのマウントを抑制すると効率的である。

【００９９】

図２０は、マウントを抑制する設定ファイルの一例の構成図である。図２０のような設定ファイルの場合、プログラム起動部５２は１行目のｍｏｕｎｔコマンドで指定されたチェック処理「ｍｅｍｃｈｅｃｋ３」を起動する。例えば図２０のチェック処理「ｍｅｍｃｈｅｃｋ３」は、システムメモリ６２及びローカルメモリ６７のメモリサイズが６４ＭＢ以上のときに正常終了するものとする。

【０１００】

プログラム起動部５２は、チェック処理が正常終了するとｍｏｕｎｔコマンドで指定された「ｗｅｂ．ｒｏｍｆｓ」をディレクトリ「／ｗｅｂ」へマウントする一方、チェック処理が異常終了するとマウントしない。例えば図２０の設定ファイルの記述の下に図６のような記述があれば、ディレクトリ「／ｗｅｂ」以下のプログラム「／ｗｅｂ／ｂｉｎ／ｈｔｔｐｄ」なども起動されなくなる。したがって、プログラム起動部５２はシステムメモリ６２及びローカルメモリ６７のメモリサイズに応じて無駄にメモリを消費することを避けることができる。

【０１０１】

なお、図２０の設定ファイルによるマウント処理は、図５のステップＳ１１で行われる処理である。また、図２０の設定ファイルでは、メモリサイズに応じてディレクトリのマウントを抑制する例について表しているが、ハードウェア資源の有無やＣＰＵ性能などに

応じてディレクトリのマウントを抑制するようにしてもよい。

【実施例３】

【０１０２】

図１８は、プログラム起動部５２のチェック処理と本体プログラムとを１対ｎに対応させた例を表しているが、図２１のようにチェック処理と本体プログラムとをｎ対１に対応させるようにしてもよい。

【０１０３】

図２１は、チェック処理と本体プログラムとがｎ対１に対応する例について説明するための図である。図２１の例では、プログラム起動部５２のチェック処理ａ，ｂと本体プログラムａとが対応している。プログラム起動部５２はチェック処理ａ，ｂを起動し、チェック処理ａ，ｂが正常終了したとき、チェック処理ａ，ｂに対応している本体プログラムａを起動する。図２１のようにチェック処理ａ，ｂと本体プログラムａとが対応している場合、設定ファイルは図２２のように構成される。図２２は、設定ファイルの他の一例の構成図である。

【０１０４】

図２２の設定ファイルでは、１つのｅｘｅｃコマンドに「－ｈ」オプションのある２つのチェック処理「チェック処理ａ」及び「チェック処理ｂ」が設定されている。したがって、プログラム起動部５２は図５のステップＳ１１３でチェック処理ａ，ｂを実行する。そして、プログラム起動部５２はステップＳ１１４でチェック処理ａ，ｂが両方とも正常終了したか否かを判定し、両方とも正常終了したと判定したときにｅｘｅｃコマンドで指定された本体プログラムａを起動する。なお、プログラム起動部５２のステップＳ１１３及びＳ１１４以外の処理は図５と同様であるので説明を省略する。

【実施例４】

【０１０５】

融合機１では、設定ファイルに応じて、同一のチェック処理が繰り返し実行されることもある。したがって、以前に実行したチェック処理を再び実行する場合は、そのチェック処理のチェックの結果を利用する方が処理時間の短縮の観点から望ましい。そこで、本発明の融合機１は図２３及び図２４のような処理を行うことにより、以前に起動されたチェック処理のチェックの結果を利用できるようにする。

【０１０６】

図２３及び図２４は、プログラム起動部が行うチェック処理の一例のフローチャートである。なお、ステップＳ１９０～Ｓ１９３までの処理は図５のステップＳ１１０～Ｓ１１３の処理と同様であり、説明を省略する。

【０１０７】

ステップＳ１９４では、プログラム起動部５２が、所定のメモリ領域にチェックの結果が格納されているか否かで既にチェック済みであるか否かを判定する。例えばチェックの結果を格納する所定のメモリ領域には、ＯＳを介することなくプログラム起動部５２のプロセスが直接アクセス可能なメモリ上の領域を利用できる。チェック済みであると判定すると（Ｓ１９４においてＹＥＳ）、プログラム起動部５２はステップＳ１９５に進み、所定のメモリ領域からチェックの結果を読み出したあとステップＳ１９８に進む。一方、チェック済みでないと判定すると（Ｓ１９４においてＮＯ）、プログラム起動部５２はステップＳ１９６に進み、前述したようなハードウェア資源に関するチェックを行う。

【０１０８】

ステップＳ１９６に続いてステップＳ１９７に進み、プログラム起動部５２はチェックの結果を所定のメモリ領域に書き込んだあとステップＳ１９８に進む。なお、ステップＳ１９８～Ｓ２００の処理は、図５のステップＳ１１４～Ｓ１１６の処理と同様であり、説明を省略する。図２３および図２４のフローチャートにより、以前に起動されたチェック処理のチェックの結果を利用できる。

【０１０９】

さらに、融合機１は起動時に全てのチェック処理を実行し、全てのチェック処理による

チェックの結果を所定のメモリ領域に予め書き込んでおいてもよい。この場合、融合機１は起動直後にチェック処理によるチェックを行ったあと、所定のメモリ領域に書き込まれたチェックの結果を利用することができるので、処理時間の短縮が可能となる。

【０１１０】

本発明は、具体的に開示された実施例に限定されるものではなく、特許請求のの範囲から逸脱することなく、種々の変形や変更が可能である。

【図面の簡単な説明】

【０１１１】

【図１】本発明による融合機のソフトウェア構成について説明するための一実施例の構成図である。

【図２】本発明による融合機のハードウェア構成について説明するための一実施例の構成図である。

【図３】融合機起動部の一例の構成図である。

【図４】融合機起動部の処理の一例のフローチャートである。

【図５】プログラム起動部の処理の一例のフローチャートである。

【図６】設定ファイルの一例の構成図である。

【図７】チェック処理「ｆｃｕｃｈｅｃｋ」の一例のフローチャートである。

【図８】チェック処理「ｃｐｕｃｈｅｃｋ１」の一例のフローチャートである。

【図９】チェック処理「ｃｐｕｃｈｅｃｋ２」の一例のフローチャートである。

【図１０】チェック処理「ｍｅｍｃｈｅｃｋ１」の一例のフローチャートである。

【図１１】チェック処理「ｍｅｍｃｈｅｃｋ２」の一例のフローチャートである。

【図１２】設定ファイルの他の一例の構成図である。

【図１３】チェック処理「ｈｄｄｎｏｎｅｘｉｓｔ」の一例のフローチャートである。

【図１４】設定ファイルの他の一例の構成図である。

【図１５】ＳＤカード内に格納されているファイルの一例のイメージ図である。

【図１６】チェック処理「ｓｄｃｏｍｍａｎｄ」の一例のフローチャートである。

【図１７】本体プログラムと，チェック処理と，プログラム起動部と，ＯＳと，ハードウェアとの関係を表した一例の関係図である。

【図１８】チェック処理と本体プログラムとが１対ｎに対応する例について説明するための図である。

【図１９】設定ファイルの他の一例の構成図である。

【図２０】マウントを抑制する設定ファイルの一例の構成図である。

【図２１】チェック処理と本体プログラムとがｎ対１に対応する例について説明するための図である。

【図２２】設定ファイルの他の一例の構成図である。

【図２３】プログラム起動部が行うチェック処理の一例のフローチャートである（１／２）。

【図２４】プログラム起動部が行うチェック処理の一例のフローチャートである（２／２）。

【符号の説明】

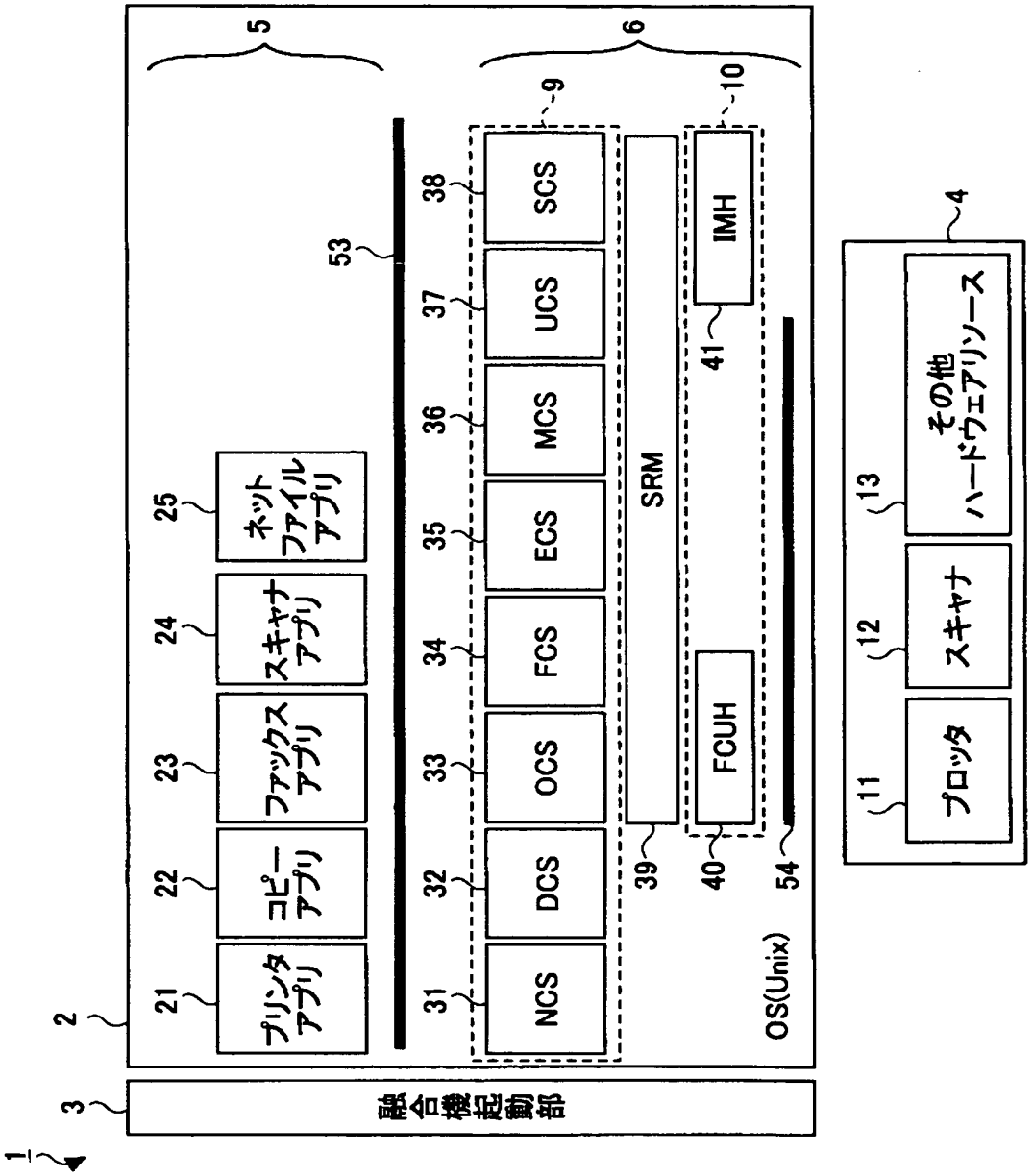
【０１１２】

１　融合機
２　ソフトウェア群
３　融合機起動部
４　ハードウェア資源
５　アプリケーション層
６　プラットフォーム
９　コントロールサービス層
１０　ハンドラ層

１１　プロッタ
１２　スキャナ
１３　ハードウェアリソース
２１　プリンタアプリ
２２　コピーアプリ
２３　ファックスアプリ
２４　スキャナアプリ
２５　ネットファイルアプリ
３１　ネットワークコントロールサービス（ＮＣＳ）
３２　デリバリーコントロールサービス（ＤＣＳ）
３３　オペレーションパネルコントロールサービス（ＯＣＳ）
３４　ファックスコントロールサービス（ＦＣＳ）
３５　エンジンコントロールサービス（ＥＣＳ）
３６　メモリコントロールサービス（ＭＣＳ）
３７　ユーザインフォメーションコントロールサービス（ＵＣＳ）
３８　システムコントロールサービス（ＳＣＳ）
３９　システムリソースマネージャ（ＳＲＭ）
４０　ファックスコントロールユニットハンドラ（ＦＣＵＨ）
４１　イメージメモリハンドラ（ＩＭＨ）
５１　ＲＯＭモニタ
５２　プログラム起動部
５３　アプリケーションプログラムインターフェース（ＡＰＩ）
５４　エンジンＩ／Ｆ
６０　コントローラ
６１　ＣＰＵ
６２　システムメモリ
６３　ノースブリッジ（ＮＢ）
６４　サウスブリッジ（ＳＢ）
６５　ＡＧＰ（Accelerated Graphics Port）
６６　ＡＳＩＣ
６７　ローカルメモリ
６８　ハードディスク装置（ＨＤＤ）
６９　ネットワークインターフェースコントローラ（ＮＩＣ）
７０　ＳＤカード用スロット
７１　ＵＳＢ　Ｉ／Ｆ
７２　ＩＥＥＥ１３９４　Ｉ／Ｆ
７３　セントロニクス　Ｉ／Ｆ
８０　操作パネル
８１　ファックスコントロールユニット（ＦＣＵ）
８２　エンジン部
８３　ＰＣＩバス

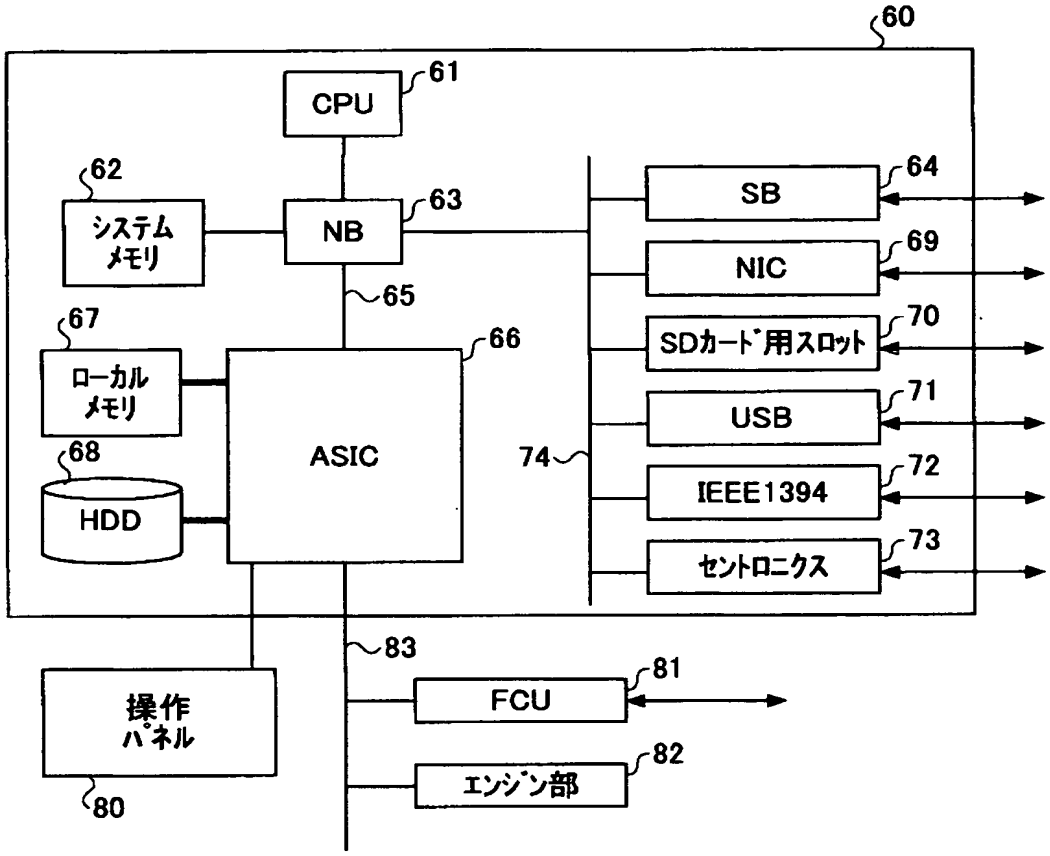
【書類名】図面
【図１】

本発明による融合機のソフトウェア構成について説明するための一実施例の構成図



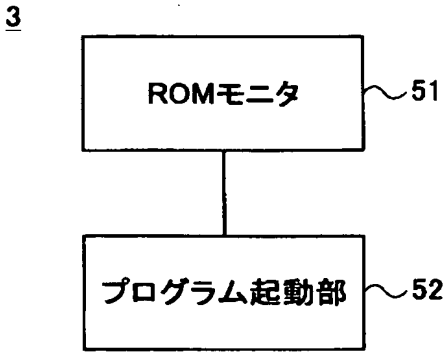
【図２】

本発明による融合機のハードウェア構成について説明するための一実施例の構成図



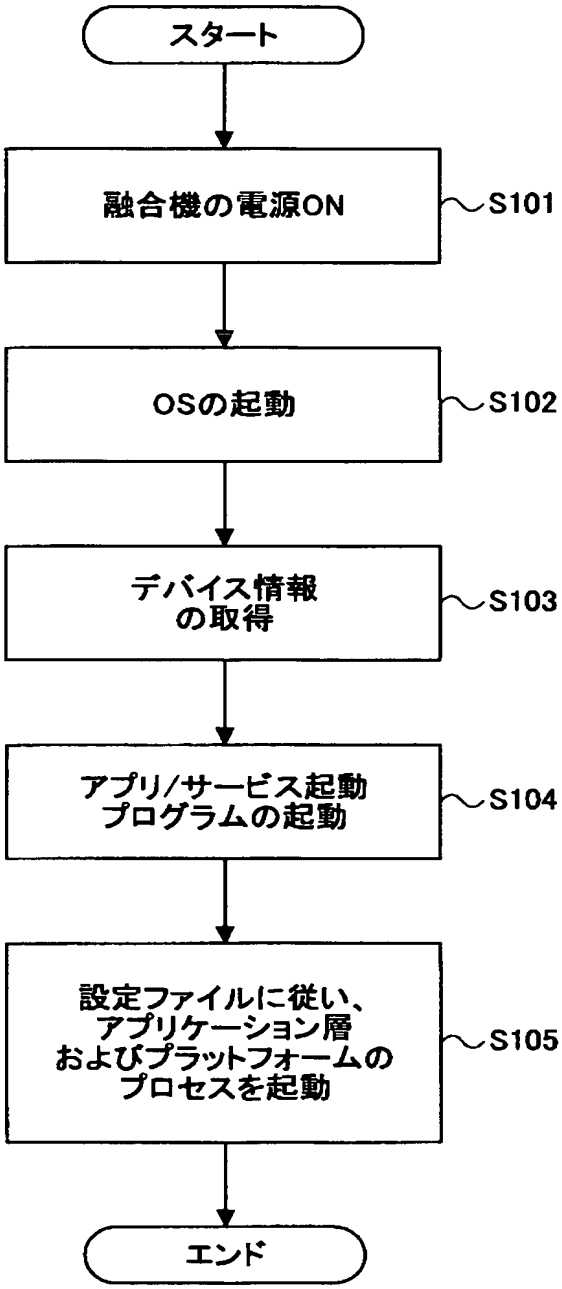
【図３】

融合機起動部の一例の構成図



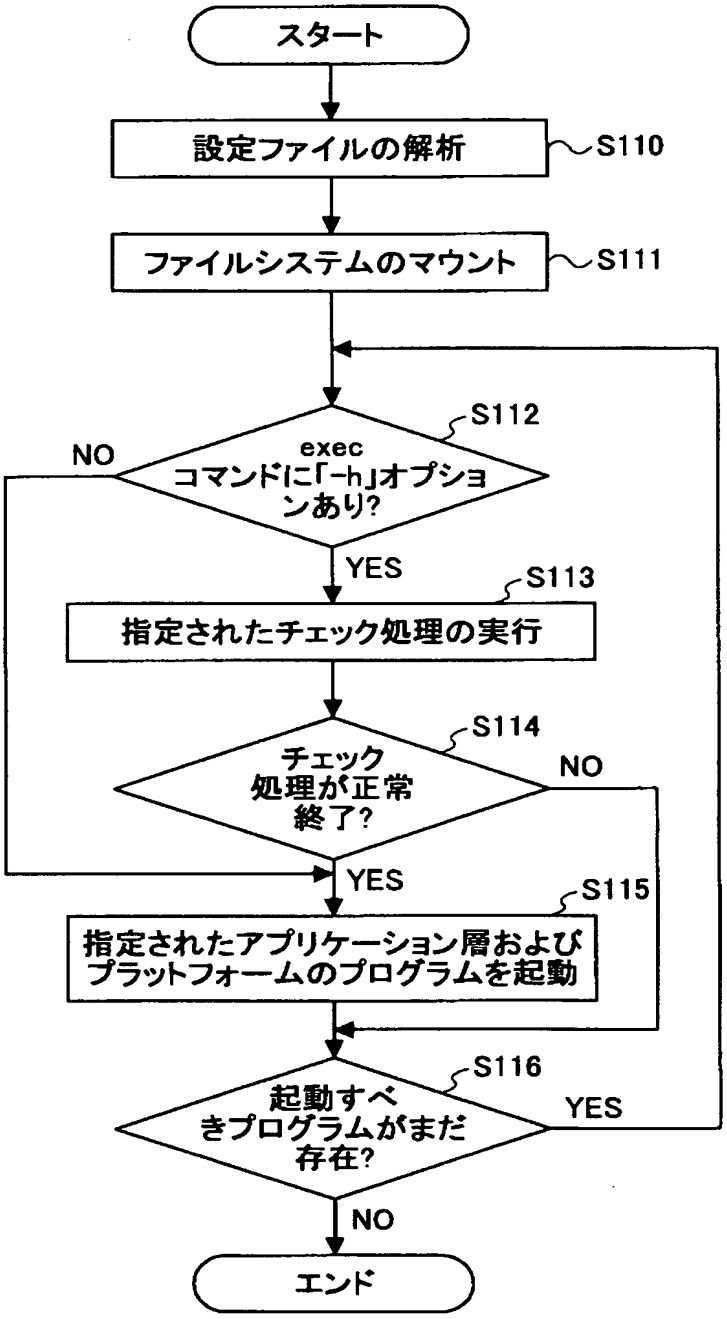
【図４】

融合機起動部の処理の一例のフローチャート



【図５】

プログラム起動部の処理の一例のフローチャート



【図６】

設定ファイルの一例の構成図

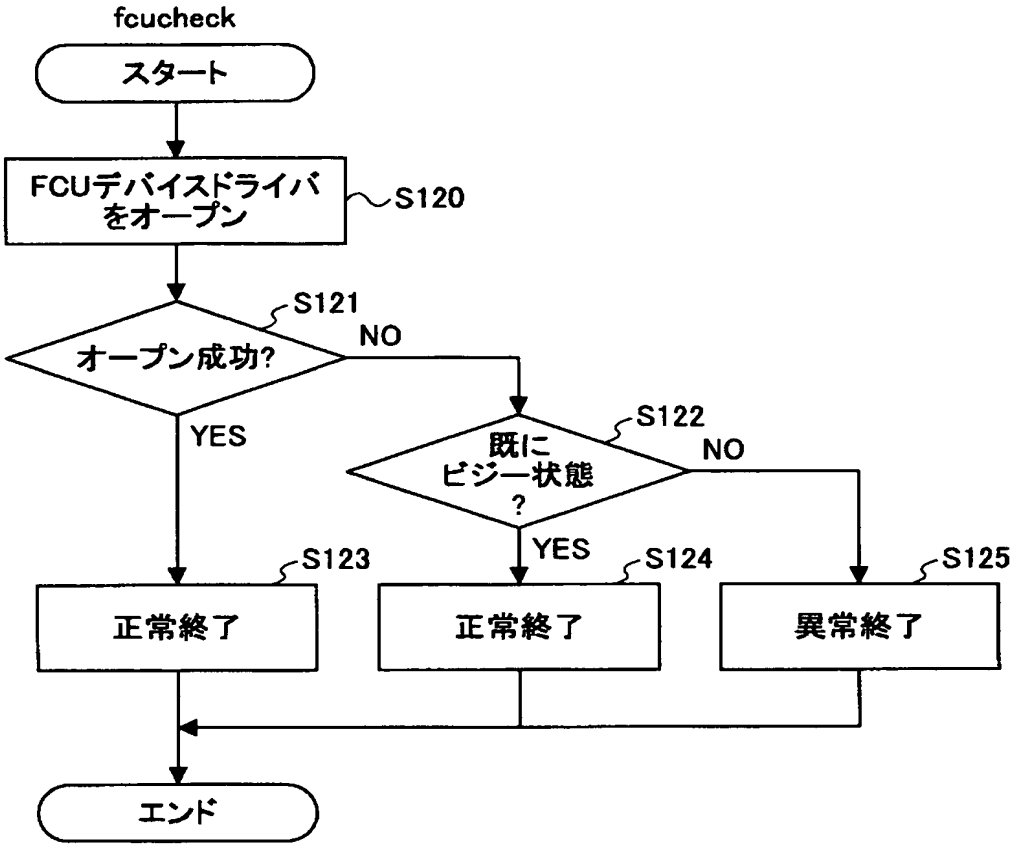
```
exec –h fcucheck /fax/bin/fax

exec –h cpucheck1 /printer/bin/setfont_bitmap
exec –h cpucheck2 /printer/bin/setfont_vector

exec –h memcheck1 /web/bin/httpd –5
exec –h memcheck2 /web/bin/httpd –10
```

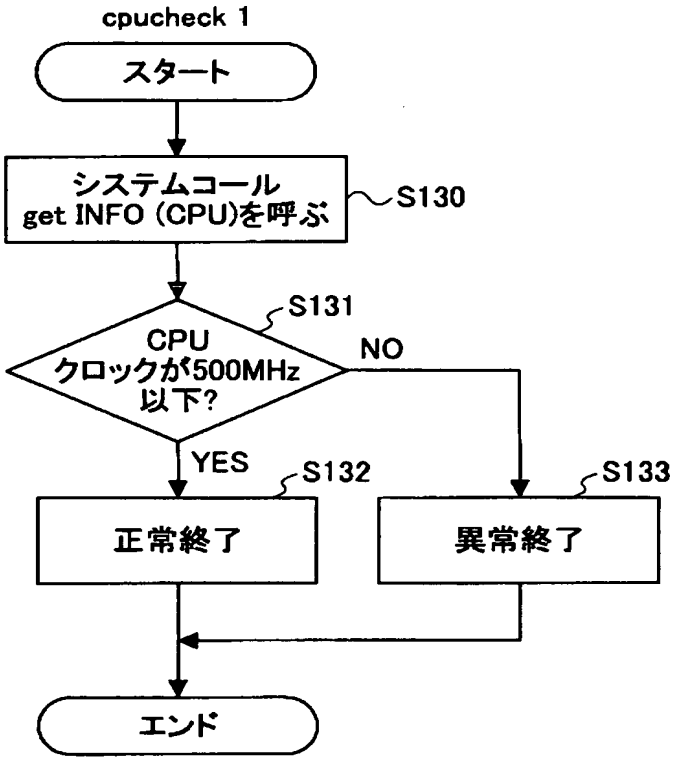
【図７】

チェック処理「ｆｃｕｃｈｅｃｋ」の一例のフローチャート



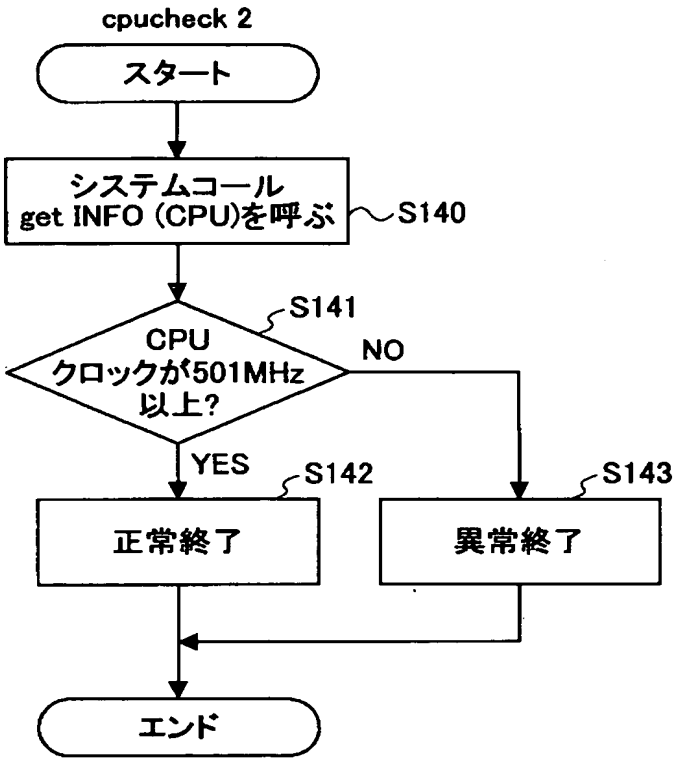
【図８】

チェック処理「ｃｐｕｃｈｅｃｋ１」の一例のフローチャート



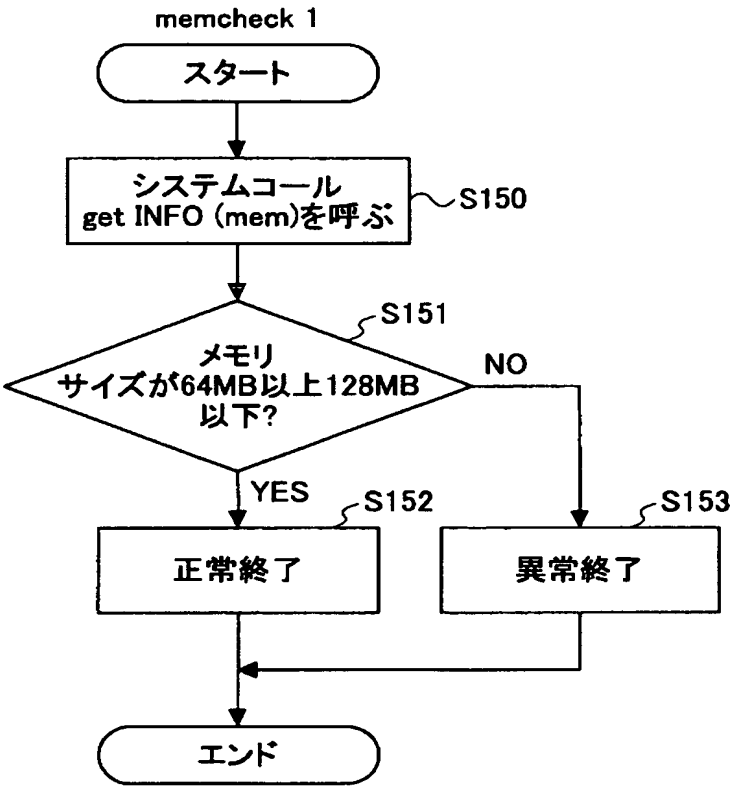
【図９】

チェック処理「ｃｐｕｃｈｅｃｋ２」の一例のフローチャート



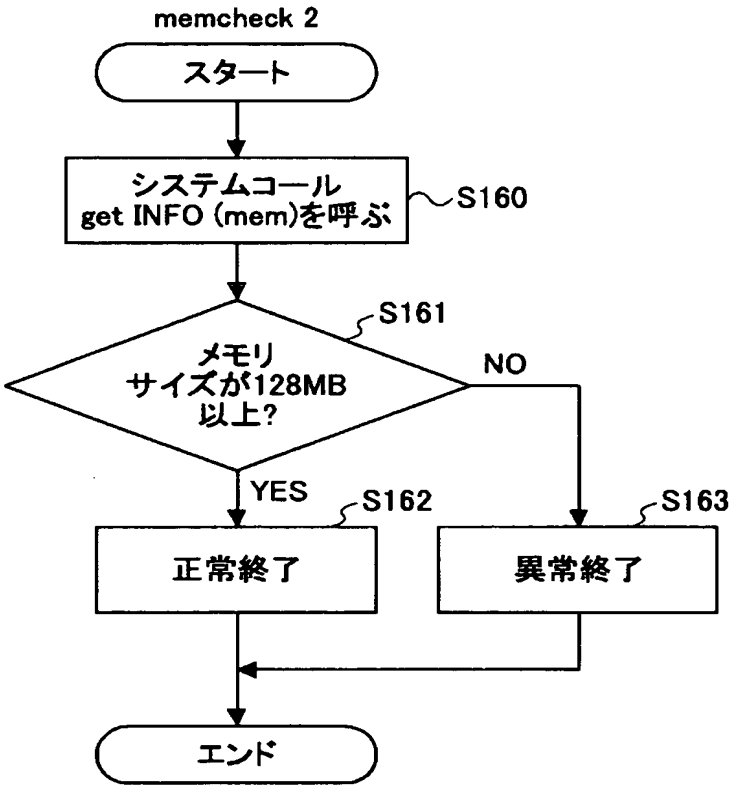
【図１０】

チェック処理「ｍｅｍｃｈｅｃｋ１」の一例のフローチャート



【図１１】

チェック処理「ｍｅｍｃｈｅｃｋ２」の一例のフローチャート



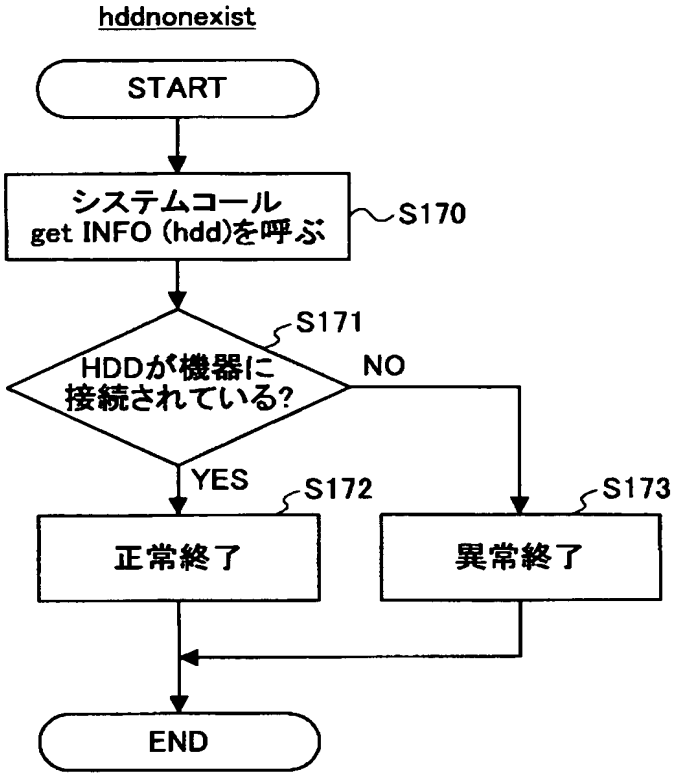
【図１２】

設定ファイルの他の一例の構成図

```
mount -h hddnonexist ramdisk /dev/md0c /ramdisk
```

【図１３】

チェック処理「hddnonexist」の一例のフローチャート



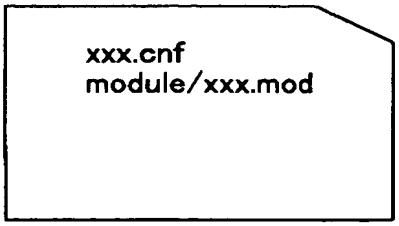
【図１４】

設定ファイルの他の一例の構成図

```
sd2
mount -h /sbin/sdcommand gzromfs xxx.mod /mnt
exec /mnt/xxx
```

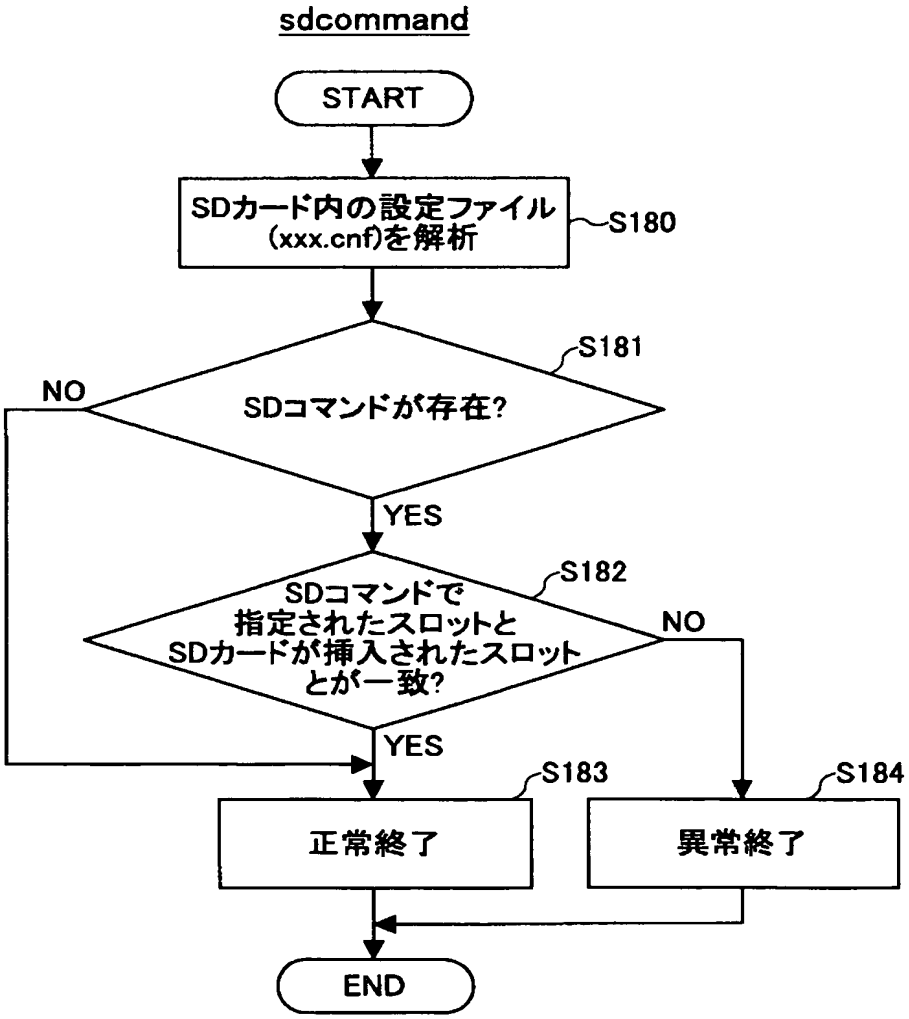
【図１５】

SDカード内に格納されているファイルの一例のイメージ図



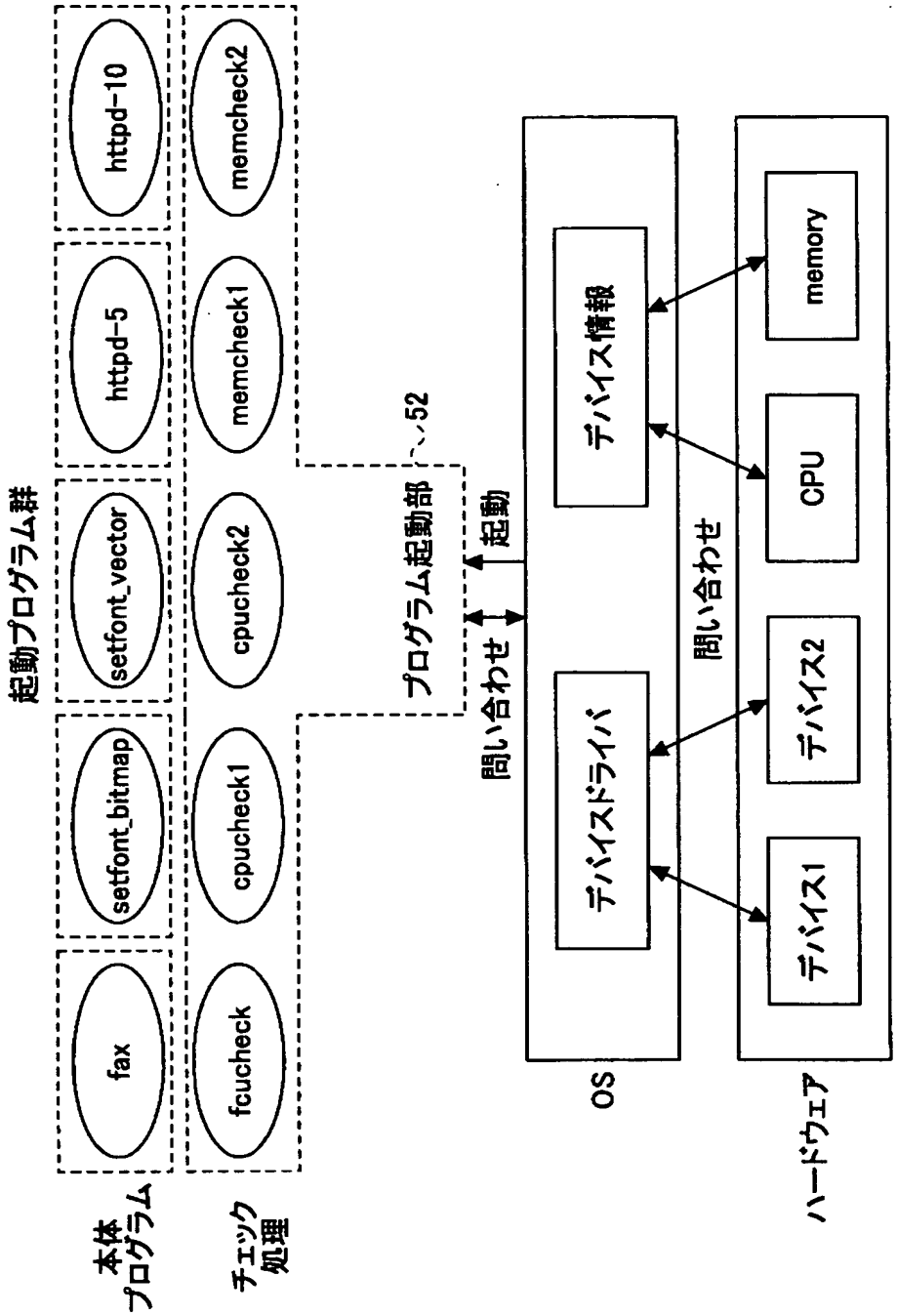
【図１６】

チェック処理「sdcommand」の一例のフローチャート



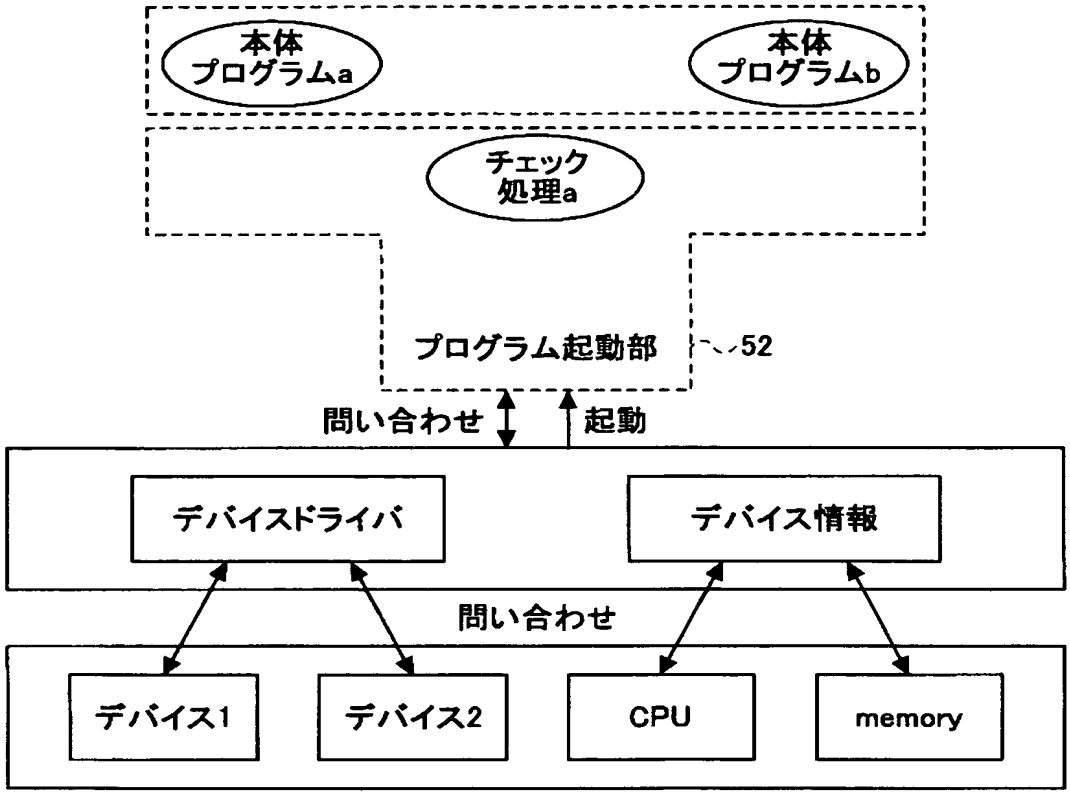
【図１７】

本体プログラムと，チェック処理と，プログラム起動部と，
ＯＳと，ハードウェアとの関係を表した一例の関係図



【図１８】

チェック処理と本体プログラムとが
１対ｎに対応する例について説明するための図



【図１９】

設定ファイルの他の一例の構成図

exec –h [ﾁｪｯｸ処理a] [本体ﾌﾟﾛｸﾞﾗﾑa]

exec –h [ﾁｪｯｸ処理a] [本体ﾌﾟﾛｸﾞﾗﾑb]

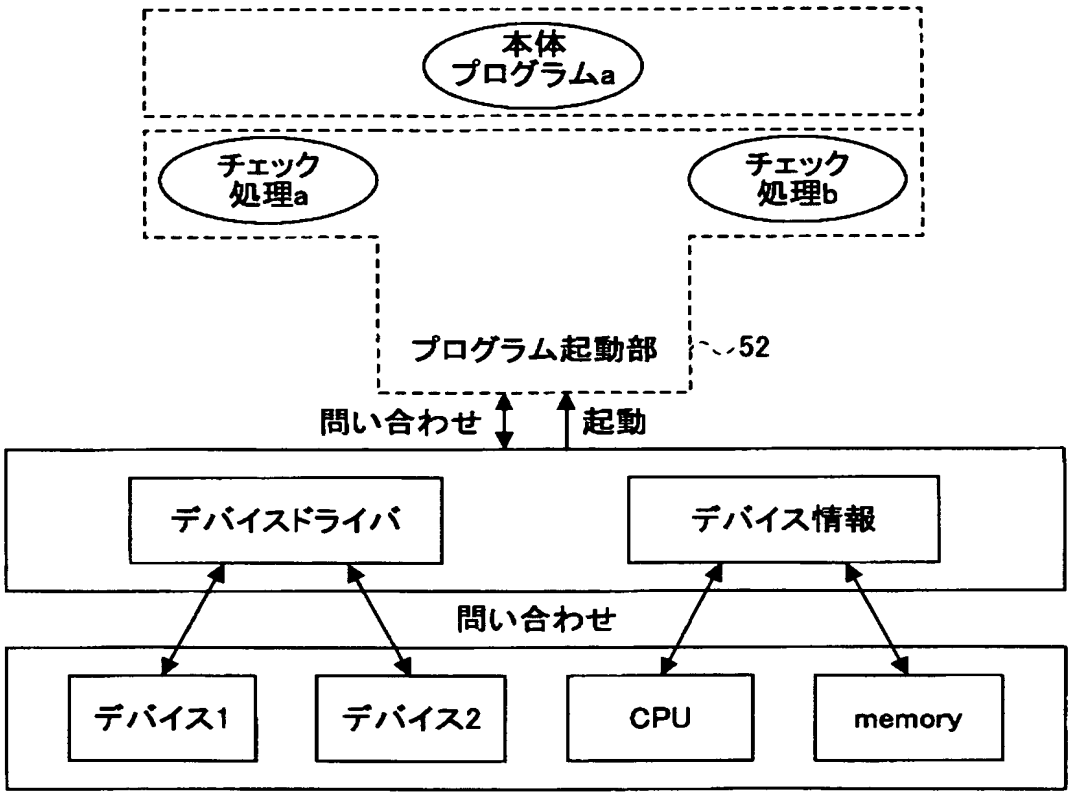
【図２０】

マウントを抑制する設定ファイルの一例の構成図

```
mount -h memcheck3 romfs web.romfs /web
```

【図２１】

チェック処理と本体プログラムとが
ｎ対１に対応する例について説明するための図



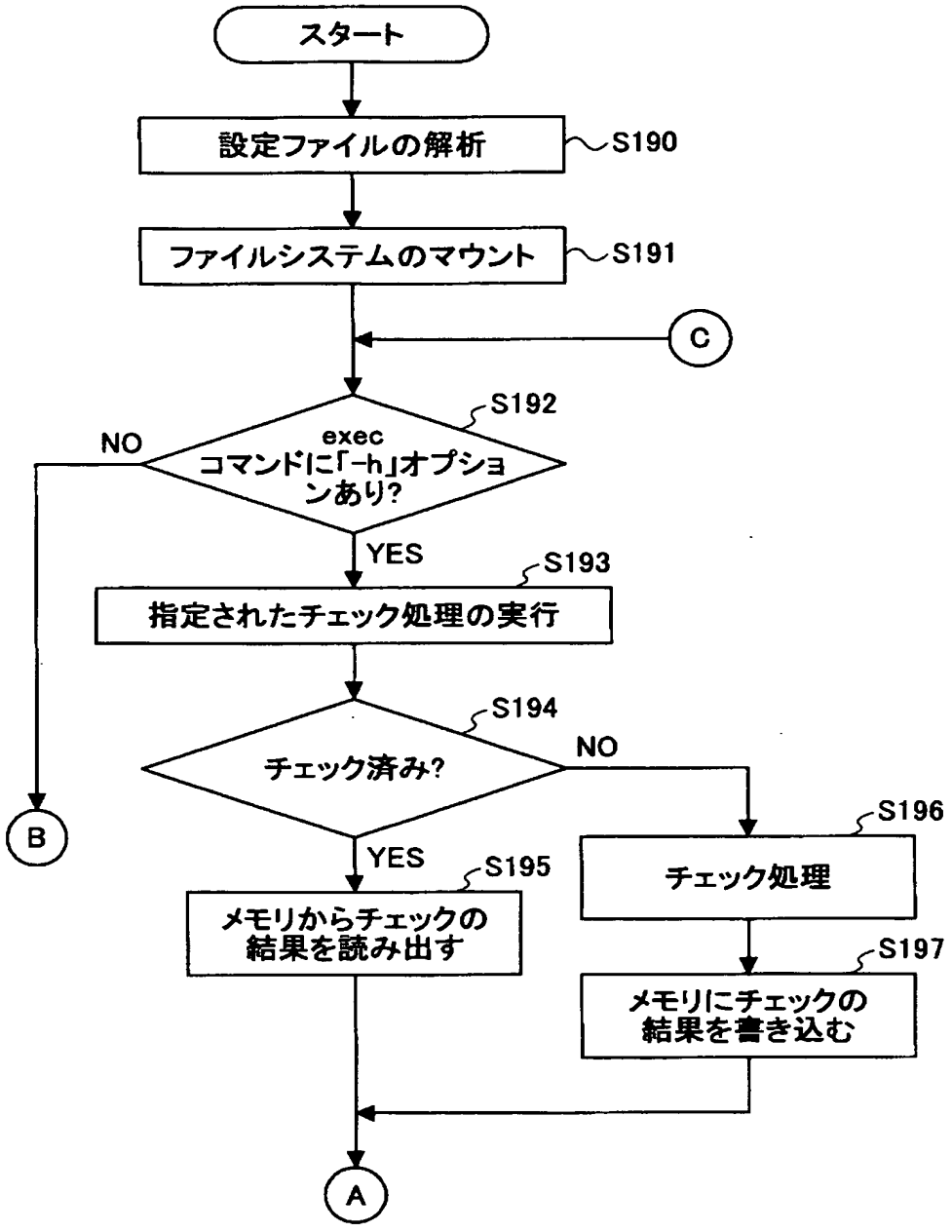
【図２２】

設定ファイルの他の一例の構成図

exec –h [ﾁｪｯｸ処理a]–h [ﾁｪｯｸ処理b] [本体ﾌﾟﾛｸﾞﾗﾑa]

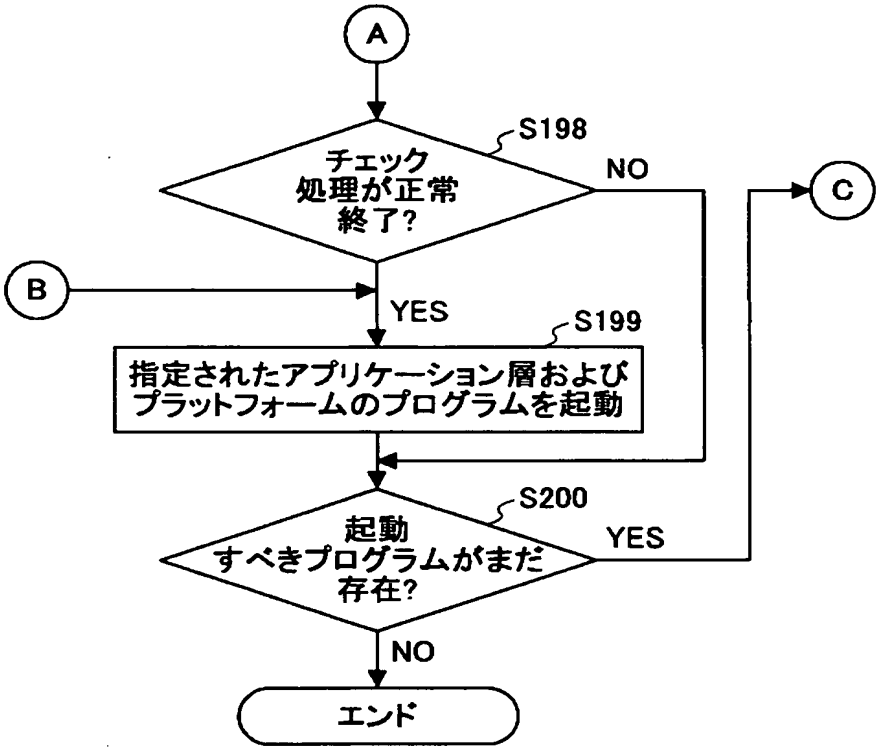
【図２３】

プログラム起動部が行うチェック処理の一例のフローチャート（１／２）



【図２４】

プログラム起動部が行うチェック処理の
一例のフローチャート（２／２）



【書類名】要約書
【要約】
【課題】　各プログラムの重複部分を削減することができ、ハードウェア資源に関連するプログラムを効率良く起動することが可能な画像形成装置およびプログラム起動方法を提供することを目的とする。
【解決手段】　画像形成処理で使用されるハードウェア資源と、画像形成に係る処理を行うプログラムとを有する画像形成装置であって、ハードウェア資源に関するチェックを行うチェック処理とプログラムとの関連を設定する設定手段と、チェック処理を行い、そのチェック処理の結果に応じてチェック処理に関連するプログラムを起動する起動手段５２とを有することにより上記課題を解決する。
【選択図】　図１７

特願２００３－３９３４１６

出　願　人　履　歴　情　報

識別番号　　［０００００６７４７］

１．変更年月日　　２００２年　５月１７日
［変更理由］　　住所変更
住　所　　東京都大田区中馬込１丁目３番６号
氏　名　　株式会社リコー